```
import re

if re.search("inform", "we need to inform him with the latest information"):
    print("There is inform")
else:
    print("Not found")


if re.search("must", "we need to inform him with the latest information"):
    print("Must is there")
else:
    print("Not found")#  Generating an iterator:
#
# Generating an iterator is the simple process of finding out and reporting the starting and the ending index of the string. Consider the following example:

import re

Str = "we need to inform him with the latest information"

for i in re.finditer("inform.", Str):
    locTuple = i.span()
    print(locTuple)
# For every match found, the starting and the ending index is printed. Can you take a guess of the output that we get when we execute the above program? Check it out below .# Matching words with patterns:
#
# Consider an input string where you have to match certain words with the string. To elaborate, check out the following example code:
#

import re

Str = "Sat, hat, mat, pat"

allStr = re.findall("[shmp]at", Str)

for i in allStr:
    print(i)#Matching series of range of characters:


import re

Str = "sat, hat, mat, pat"

someStr = re.findall("[h-m]at", Str)

for i in someStr:
    print(i)# We have added a caret symbol(^) in the Regular Expression. What this does  it negates the effect of whatever it follows. Instead of giving us the output of everything starting with h to m, we will be presented with the output of everything apart from that.
import re

Str = "sat, hat, mat, pat"

someStr = re.findall("[^h-m]at", Str)

for i in someStr:
```

```
    print(i)import re

Food = "hat rat mat pat at"

regex = re.compile("[r]at")

Food = regex.sub("food", Food)

print(Food)import re

randstr = "12345"

print("\d Matches:", len(re.findall("\d", randstr)))



# As you can see from the above output, d matches the integers present in the string. However if we replace it with D, it will match everything BUT an integer, the exact opposite of d.

randstr = "12345"

print(" \\D Matches:", len(re.findall("\D", randstr)))import re

randstr = '''
Delhi is
capital of
India
'''

print(randstr)

regex = re.compile("\n")

randstr = regex.sub(" ", randstr)

print(randstr)# Practical Use Cases Of Regular Expressions
# # We will be checking out 3 main use-cases which are widely used on a daily basis. Following are the concepts we will be checking out:
# #
# # Phone Number Verification
# # E-mail Address Verification
# # Web Scraping
# # Let us begin this section of Python RegEx tutorial by checking out the first case.
# #
# # Phone Number Verification:
# #
# # Problem Statement - The need to easily verify phone numbers in any relevant scenario.
# #
# # Consider the following Phone numbers:
# #
# # 444-122-1234
# # 123-122-78999
# # 111-123-23
# # 67-7890-2019
# # The general format of a phone number is as follows:
# #
# # Starts with 3 digits and '-' sign
```

```
108 # # 3 middle digits and '-' sign
109 # # 4 digits in the end
110
111 import re
112
113 phn = "412-555-1212"
114
115 if re.search("\d{3}-\d{3}-\d{4}", phn):
116     print("Valid phone number")
117 else:
118     print("Invalid Phone Number")import urllib.request
119 from re import findall
120
121 url = "http://www.summet.com/dmsi/html/codesamples/addresses.html"
122
123 response = urllib.request.urlopen(url)
124
125 html = response.read()
126
127 htmlStr = html.decode()
128 print(htmlStr)
129 pdata = findall("\([1-9]{3}\) \d{3}-\d{4}", htmlStr)
130
131 for item in pdata:
132     print(item)# \   Used to drop the special meaning of character
133 #     following it (discussed below)
134 # []  Represent a character class
135 # ^   Matches the beginning
136 # $   Matches the end
137 # .   Matches any character except newline
138 # ?   Matches zero or one occurrence.
139 # |   Means OR (Matches with any of the characters
140 #     separated by it.
141 # *   Any number of occurrences (including 0 occurrences)
142 # +   One ore more occurrences
143 # {}  Indicate number of occurrences of a preceding RE
144 #     to match.
145 # ()  Enclose a group of REs
146 # \d   Matches any decimal digit, this is equivalent
147 #      to the set class [0-9].
148 # \D   Matches any non-digit character.
149 # \s   Matches any whitespace character.
150 # \S   Matches any non-whitespace character
151 # \w   Matches any alphanumeric character, this is
152 #      equivalent to the class [a-zA-Z0-9_].
153 # \W   Matches any non-alphanumeric character.
154
155 # Module Regular Expression is imported using __import__().
156 import re
157
158 # compile() creates regular expression character class [a-e],
159 # which is equivalent to [abcde].
160 # class [abcde] will match with string with 'a', 'b', 'c', 'd', 'e'.
161 p = re.compile('[a-e]')
162
163 # findall() searches for the Regular Expression and return a list upon finding
164 print(p.findall("Aye, said Mr. Gibenson Stark"))
165 import re
166
```

```
167 # \d is equivalent to [0-9].
168 p = re.compile('\d')
169 print(p.findall("I went to him at 11 A.M. on 4th July 1886"))
170
171 # \d+ will match a group on [0-9], group of one or greater size
172 p = re.compile('\d+')
173 print(p.findall("I went to him at 11 A.M. on 4th July 1886"))
174 import re
175
176 # \w is equivalent to [a-zA-Z0-9_].
177 p = re.compile('\w')
178 print(p.findall("He said * in some_lang."))
179
180 # \w+ matches to group of alphanumeric character.
181 p = re.compile('\w+')
182 print(p.findall("I went to him at 11 A.M., he said *** in some_language."))
183
184 # \W matches to non alphanumeric characters.
185 p = re.compile('\W')
186 print(p.findall("he said *** in some_language."))
187 import re
188
189 # '*' replaces the no. of occurrence of a character.
190 p = re.compile('ab*')
191 print(p.findall("ababbaabbb"))
192 from re import split
193
194 # '\W+' denotes Non-Alphanumeric Characters or group of characters
195 # Upon finding ',' or whitespace ' ', the split(), splits the string from that point
196 print(split('\W+', 'Words, words , Words'))
197 print(split('\W+', "Word's words Words"))
198
199 # Here ':', ' ' ,',' are not AlphaNumeric thus, the point where splitting occurs
200 print(split('\W+', 'On 12th Jan 2016, at 11:02 AM'))
201
202 # '\d+' denotes Numeric Characters or group of characters
203 # Spliting occurs at '12', '2016', '11', '02' only
204 print(split('\d+', 'On 12th Jan 2016, at 11:02 AM'))
205 import re
206
207 # Splitting will occurs only once, at '12', returned list will have length 2
208 print(re.split('\d+', 'On 12th Jan 2016, at 11:02 AM', 1))
209
210 # 'Boy' and 'boy' will be treated same when flags = re.IGNORECASE
211 print(re.split('[a-f]+', 'Aey, Boy oh boy, come here', flags = re.IGNORECASE))
212 print(re.split('[a-f]+', 'Aey, Boy oh boy, come here'))
213 import re
214
215 # Regular Expression pattern 'ub' matches the string at "Subject" and "Uber".
216 # As the CASE has been ignored, using Flag, 'ub' should match twice with the string
217 # Upon matching, 'ub' is replaced by '~*' in "Subject", and in "Uber", 'Ub' is
    replaced.
218 print(re.sub('ub', '~*' , 'Subject has Uber booked already', flags = re.IGNORECASE))
219
220 # Consider the Case Senstivity, 'Ub' in "Uber", will not be reaplced.
221 print(re.sub('ub', '~*' , 'Subject has Uber booked already'))
222
223 # As count has been given value 1, the maximum times replacement occurs is 1
224 print(re.sub('ub', '~*' , 'Subject has Uber booked already', count=1, flags = re.
```

```
224 IGNORECASE))
225
226 # 'r' before the patter denotes RE, \s is for start and end of a String.
227 print(re.sub(r'\sAND\s', ' & ', 'Baked Beans And Spam', flags=re.IGNORECASE))
228 # Python3 program to extract all the numbers from a string
229 import re
230
231 # Function to extract all the numbers from the given string
232 def getNumbers(str):
233     array = re.findall(r'[0-9]+', str)
234     return array
235
236 # Driver code
237 str = "adbv345hj43hvb42"
238 array = getNumbers(str)
239 print(*array)
240 # $ Matches the end of the line
241 # \s    Matches whitespace
242 # \S    Matches any non-whitespace character
243 # * Repeats a character zero or more times
244 # \S    Matches any non-whitespace character
245 # *?    Repeats a character zero or more times (non-greedy)
246 # + Repeats a character one or more times
247 # +?    Repeats a character one or more times (non-greedy)
248 # [aeiou]   Matches a single character in the listed set
249 # [^XYZ]    Matches a single character not in the listed set
250 # [a-z0-9]  The set of characters can include a range
251 # ( Indicates where string extraction is to start
252 # ) Indicates where string extraction is to end
253
254 import re
255
256 # Example string
257 s = 'Hello from shubhamg199630@gmail.com to priya@yahoo.com about the meeting @2PM'
258
259 # \S matches any non-whitespace character
260 # @ for as in the Email
261 # + for Repeats a character one or more times
262 lst = re.findall('\S+@\S+', s)
263
264 # Printing of List
265 print(lst)# A Python program to demonstrate working of re.match().
266 import re
267
268 # Lets use a regular expression to match a date string
269 # in the form of Month name followed by day number
270 regex = r"([a-zA-Z]+) (\d+)"
271
272 match = re.search(regex, "I was born on June 24")
273
274 if match != None:
275
276     # We reach here when the expression "([a-zA-Z]+) (\d+)"
277     # matches the date string.
278
279     # This will print [14, 21), since it matches at index 14
280     # and ends at 21.
281     print("Match at index %s, %s" % (match.start(), match.end()))
282
```

```
283     # We us group() method to get all the matches and
284     # captured groups. The groups contain the matched values.
285     # In particular:
286     # match.group(0) always returns the fully matched string
287     # match.group(1) match.group(2), ... return the capture
288     # groups in order from left to right in the input string
289     # match.group() is equivalent to match.group(0)
290 
291     # So this will print "June 24"
292     print("Full match: %s" % (match.group(0)))
293 
294     # So this will print "June"
295     print("Month: %s" % (match.group(1)))
296 
297     # So this will print "24"
298     print("Day: %s" % (match.group(2)))
299 
300 else:
301     print("The regex pattern does not match.")
302 # A Python program to demonstrate working
303 # of re.match().
304 # re.match() : This function attempts to match pattern to whole string. The re.match
    function returns a match object on success, None on failure.
305 import re
306 
307 
308 # a sample function that uses regular expressions
309 # to find month and day of a date.
310 def findMonthAndDate(string):
311     regex = r"([a-zA-Z]+) (\d+)"
312     match = re.match(regex, string)
313 
314     if match == None:
315         print
316         "Not a valid date"
317         return
318 
319     print
320     "Given Data: %s" % (match.group())
321     print
322     "Month: %s" % (match.group(1))
323     print
324     "Day: %s" % (match.group(2))
325 
326 
327 # Driver Code
328 findMonthAndDate("Jun 24")
329 print("")
330 findMonthAndDate("I was born on June 24")
331 #re.findall() : Return all non-overlapping matches of pattern in string, as a list of
    strings. The string is scanned left-to-right, and matches are returned in the order
    found (Source : Python Docs).
332 
333 # A Python program to demonstrate working of
334 # findall()
335 import re
336 
337 # A sample text string where regular expression
338 # is searched.
```

```
339 string = """Hello my Number is 123456789 and
340             my friend's number is 987654321"""
341
342 # A sample regular expression to find digits.
343 regex = '\d+'
344
345 match = re.findall(regex, string)
346 print(match)
347
348 # This example is contributed by Ayush Saluja.
349
350
351 #End of Program#
352
353
354 #End of Program#
355
356 import re
357
358 if re.search("inform", "we need to inform him with the latest information"):
359     print("There is inform")
360 else:
361     print("Not found")
362
363
364 if re.search("must", "we need to inform him with the latest information"):
365     print("Must is there")
366 else:
367     print("Not found")
368 #End of Program#
369
370 #  Generating an iterator:
371 #
372 # Generating an iterator is the simple process of finding out and reporting the
    starting and the ending index of the string. Consider the following example:
373
374 import re
375
376 Str = "we need to inform him with the latest information"
377
378 for i in re.finditer("inform.", Str):
379     locTuple = i.span()
380     print(locTuple)
381 # For every match found, the starting and the ending index is printed. Can you take a
    guess of the output that we get when we execute the above program? Check it out below.
382 #End of Program#
383
384 # Matching words with patterns:
385 #
386 # Consider an input string where you have to match certain words with the string. To
    elaborate, check out the following example code:
387 #
388
389 import re
390
391 Str = "Sat, hat, mat, pat"
392
393 allStr = re.findall("[shmp]at", Str)
394
```

```
395 for i in allStr:
396     print(i)
397 #End of Program#
398
399 #Matching series of range of characters:
400
401
402 import re
403
404 Str = "sat, hat, mat, pat"
405
406 someStr = re.findall("[h-m]at", Str)
407
408 for i in someStr:
409     print(i)
410 #End of Program#
411
412 # We have added a caret symbol(^) in the Regular Expression. What this does it negates
     the effect of whatever it follows. Instead of giving us the output of everything
    starting with h to m, we will be presented with the output of everything apart from
    that.
413 import re
414
415 Str = "sat, hat, mat, pat"
416
417 someStr = re.findall("[^h-m]at", Str)
418
419 for i in someStr:
420     print(i)
421 #End of Program#
422
423 import re
424
425 Food = "hat rat mat pat at"
426
427 regex = re.compile("[r]at")
428
429 Food = regex.sub("food", Food)
430
431 print(Food)
432 #End of Program#
433
434 import re
435
436 randstr = "12345"
437
438 print("\d Matches:", len(re.findall("\d", randstr)))
439
440
441
442 # As you can see from the above output, d matches the integers present in the string.
    However if we replace it with D, it will match everything BUT an integer, the exact
    opposite of d.
443
444 randstr = "12345"
445
446 print(" \\D Matches:", len(re.findall("\D", randstr)))
447 #End of Program#
448
```

```
449 import re
450
451 randstr = '''
452 Delhi is
453 capital of
454 India
455 '''
456
457 print(randstr)
458
459 regex = re.compile("\n")
460
461 randstr = regex.sub(" ", randstr)
462
463 print(randstr)
464 #End of Program#
465
466 # Practical Use Cases Of Regular Expressions
467 # # We will be checking out 3 main use-cases which are widely used on a daily basis. Following are the concepts we will be checking out:
468 # #
469 # # Phone Number Verification
470 # # E-mail Address Verification
471 # # Web Scraping
472 # # Let us begin this section of Python RegEx tutorial by checking out the first case.
473 # #
474 # # Phone Number Verification:
475 # #
476 # # Problem Statement - The need to easily verify phone numbers in any relevant scenario.
477 # #
478 # # Consider the following Phone numbers:
479 # #
480 # # 444-122-1234
481 # # 123-122-78999
482 # # 111-123-23
483 # # 67-7890-2019
484 # # The general format of a phone number is as follows:
485 # #
486 # # Starts with 3 digits and '-' sign
487 # # 3 middle digits and '-' sign
488 # # 4 digits in the end
489
490 import re
491
492 phn = "412-555-1212"
493
494 if re.search("\d{3}-\d{3}-\d{4}", phn):
495     print("Valid phone number")
496 else:
497     print("Invalid Phone Number")
498 #End of Program#
499
500 import urllib.request
501 from re import findall
502
503 url = "http://www.summet.com/dmsi/html/codesamples/addresses.html"
504
505 response = urllib.request.urlopen(url)
```

```
506
507 html = response.read()
508
509 htmlStr = html.decode()
510 print(htmlStr)
511 pdata = findall("\([1-9]{3}\) \d{3}-\d{4}", htmlStr)
512
513 for item in pdata:
514     print(item)
515 #End of Program#
516
517 # \   Used to drop the special meaning of character
518 #     following it (discussed below)
519 # []  Represent a character class
520 # ^   Matches the beginning
521 # $   Matches the end
522 # .   Matches any character except newline
523 # ?   Matches zero or one occurrence.
524 # |   Means OR (Matches with any of the characters
525 #     separated by it.
526 # *   Any number of occurrences (including 0 occurrences)
527 # +   One ore more occurrences
528 # {}  Indicate number of occurrences of a preceding RE
529 #     to match.
530 # ()  Enclose a group of REs
531 # \d   Matches any decimal digit, this is equivalent
532 #      to the set class [0-9].
533 # \D   Matches any non-digit character.
534 # \s   Matches any whitespace character.
535 # \S   Matches any non-whitespace character
536 # \w   Matches any alphanumeric character, this is
537 #      equivalent to the class [a-zA-Z0-9_].
538 # \W   Matches any non-alphanumeric character.
539
540 # Module Regular Expression is imported using __import__().
541 import re
542
543 # compile() creates regular expression character class [a-e],
544 # which is equivalent to [abcde].
545 # class [abcde] will match with string with 'a', 'b', 'c', 'd', 'e'.
546 p = re.compile('[a-e]')
547
548 # findall() searches for the Regular Expression and return a list upon finding
549 print(p.findall("Aye, said Mr. Gibenson Stark"))
550
551 #End of Program#
552
553 import re
554
555 # \d is equivalent to [0-9].
556 p = re.compile('\d')
557 print(p.findall("I went to him at 11 A.M. on 4th July 1886"))
558
559 # \d+ will match a group on [0-9], group of one or greater size
560 p = re.compile('\d+')
561 print(p.findall("I went to him at 11 A.M. on 4th July 1886"))
562
563 #End of Program#
564
```

```
565 import re
566
567 # \w is equivalent to [a-zA-Z0-9_].
568 p = re.compile('\w')
569 print(p.findall("He said * in some_lang."))
570
571 # \w+ matches to group of alphanumeric character.
572 p = re.compile('\w+')
573 print(p.findall("I went to him at 11 A.M., he said *** in some_language."))
574
575 # \W matches to non alphanumeric characters.
576 p = re.compile('\W')
577 print(p.findall("he said *** in some_language."))
578
579 #End of Program#
580
581 import re
582
583 # '*' replaces the no. of occurrence of a character.
584 p = re.compile('ab*')
585 print(p.findall("ababbaabbb"))
586
587 #End of Program#
588
589 from re import split
590
591 # '\W+' denotes Non-Alphanumeric Characters or group of characters
592 # Upon finding ',' or whitespace ' ', the split(), splits the string from that point
593 print(split('\W+', 'Words, words , Words'))
594 print(split('\W+', "Word's words Words"))
595
596 # Here ':', ' ' ,',' are not AlphaNumeric thus, the point where splitting occurs
597 print(split('\W+', 'On 12th Jan 2016, at 11:02 AM'))
598
599 # '\d+' denotes Numeric Characters or group of characters
600 # Spliting occurs at '12', '2016', '11', '02' only
601 print(split('\d+', 'On 12th Jan 2016, at 11:02 AM'))
602
603 #End of Program#
604
605 import re
606
607 # Splitting will occurs only once, at '12', returned list will have length 2
608 print(re.split('\d+', 'On 12th Jan 2016, at 11:02 AM', 1))
609
610 # 'Boy' and 'boy' will be treated same when flags = re.IGNORECASE
611 print(re.split('[a-f]+', 'Aey, Boy oh boy, come here', flags = re.IGNORECASE))
612 print(re.split('[a-f]+', 'Aey, Boy oh boy, come here'))
613
614 #End of Program#
615
616 import re
617
618 # Regular Expression pattern 'ub' matches the string at "Subject" and "Uber".
619 # As the CASE has been ignored, using Flag, 'ub' should match twice with the string
620 # Upon matching, 'ub' is replaced by '~*' in "Subject", and in "Uber", 'Ub' is
    replaced.
621 print(re.sub('ub', '~*' , 'Subject has Uber booked already', flags = re.IGNORECASE))
622
```

```
623 # Consider the Case Senstivity, 'Ub' in "Uber", will not be reaplced.
624 print(re.sub('ub', '~*' , 'Subject has Uber booked already'))
625
626 # As count has been given value 1, the maximum times replacement occurs is 1
627 print(re.sub('ub', '~*' , 'Subject has Uber booked already', count=1, flags = re.
    IGNORECASE))
628
629 # 'r' before the patter denotes RE, \s is for start and end of a String.
630 print(re.sub(r'\sAND\s', ' & ', 'Baked Beans And Spam', flags=re.IGNORECASE))
631
632 #End of Program#
633
634 # Python3 program to extract all the numbers from a string
635 import re
636
637 # Function to extract all the numbers from the given string
638 def getNumbers(str):
639     array = re.findall(r'[0-9]+', str)
640     return array
641
642 # Driver code
643 str = "adbv345hj43hvb42"
644 array = getNumbers(str)
645 print(*array)
646
647 #End of Program#
648
649 # $ Matches the end of the line
650 # \s    Matches whitespace
651 # \S    Matches any non-whitespace character
652 # * Repeats a character zero or more times
653 # \S    Matches any non-whitespace character
654 # *?    Repeats a character zero or more times (non-greedy)
655 # + Repeats a character one or more times
656 # +?    Repeats a character one or more times (non-greedy)
657 # [aeiou]   Matches a single character in the listed set
658 # [^XYZ]    Matches a single character not in the listed set
659 # [a-z0-9]  The set of characters can include a range
660 # ( Indicates where string extraction is to start
661 # ) Indicates where string extraction is to end
662
663 import re
664
665 # Example string
666 s = 'Hello from shubhamg199630@gmail.com to priya@yahoo.com about the meeting @2PM'
667
668 # \S matches any non-whitespace character
669 # @ for as in the Email
670 # + for Repeats a character one or more times
671 lst = re.findall('\S+@\S+', s)
672
673 # Printing of List
674 print(lst)
675 #End of Program#
676
677
678 #End of Program#
679
680 # A Python program to demonstrate working of re.match().
```

```
681 import re
682
683 # Lets use a regular expression to match a date string
684 # in the form of Month name followed by day number
685 regex = r"([a-zA-Z]+) (\d+)"
686
687 match = re.search(regex, "I was born on June 24")
688
689 if match != None:
690
691     # We reach here when the expression "([a-zA-Z]+) (\d+)"
692     # matches the date string.
693
694     # This will print [14, 21), since it matches at index 14
695     # and ends at 21.
696     print("Match at index %s, %s" % (match.start(), match.end()))
697
698     # We us group() method to get all the matches and
699     # captured groups. The groups contain the matched values.
700     # In particular:
701     # match.group(0) always returns the fully matched string
702     # match.group(1) match.group(2), ... return the capture
703     # groups in order from left to right in the input string
704     # match.group() is equivalent to match.group(0)
705
706     # So this will print "June 24"
707     print("Full match: %s" % (match.group(0)))
708
709     # So this will print "June"
710     print("Month: %s" % (match.group(1)))
711
712     # So this will print "24"
713     print("Day: %s" % (match.group(2)))
714
715 else:
716     print("The regex pattern does not match.")
717
718 #End of Program#
719
720 # A Python program to demonstrate working
721 # of re.match().
722 # re.match() : This function attempts to match pattern to whole string. The re.match
    function returns a match object on success, None on failure.
723 import re
724
725
726 # a sample function that uses regular expressions
727 # to find month and day of a date.
728 def findMonthAndDate(string):
729     regex = r"([a-zA-Z]+) (\d+)"
730     match = re.match(regex, string)
731
732     if match == None:
733         print
734         "Not a valid date"
735         return
736
737     print
738     "Given Data: %s" % (match.group())
```

```
739     print
740     "Month: %s" % (match.group(1))
741     print
742     "Day: %s" % (match.group(2))
743
744
745 # Driver Code
746 findMonthAndDate("Jun 24")
747 print("")
748 findMonthAndDate("I was born on June 24")
749
750 #End of Program#
751
752 #re.findall() : Return all non-overlapping matches of pattern in string, as a list of
    strings. The string is scanned left-to-right, and matches are returned in the order
    found (Source : Python Docs).
753
754 # A Python program to demonstrate working of
755 # findall()
756 import re
757
758 # A sample text string where regular expression
759 # is searched.
760 string = """Hello my Number is 123456789 and
761             my friend's number is 987654321"""
762
763 # A sample regular expression to find digits.
764 regex = '\d+'
765
766 match = re.findall(regex, string)
767 print(match)
768
769 # This example is contributed by Ayush Saluja.
770
771
772 #End of Program#
773
774
775 #End of Program#
776
777
778 #End of Program#
779
780
```