In [ ]:

In [1]:
```
1. What is the name of the feature responsible for generating Regex objects?
Answer: re.compile() is the feature responsible for generation of Regex object

import re
x = re.compile("some_random_pattern")
type(x)
```

```
re.compile('some_random_pattern')
```

In [3]:
```
2. Why do raw strings often appear in Regex objects?
Ans: Regular expressions use the backslash character ('\') to indicate special
   characters (speical sequences) to be used without invoking their special me
  of the same character for the same purpose in string literals. Hence, Raw st
    backslashes do not have to be escaped.
```

In [4]:
```
3. What is the return value of the search() method?
Ans: The return value of re.search(pattern,string) method
     is a match object if the pattern is observed in the string else
     it returns a None

import re
match = re.search('i','Ineuron Full Stack Data Science Program', flags=re.IGNO
print('Output:',match)
match = re.search('X','Ineuron Full Stack Data Science Program', flags=re.IGNO
```

```
Output: <re.Match object; span=(0, 1), match='I'>
Output: None
```

In [5]:
```
4. From a Match item, how do you get the actual strings
that match the pattern?
Ans: For Matched items group() methods returns actual strings that match the p

import re
match = re.search('ineuron','Ineuron Full Stack Data Science Program', flags=r
```

```
Output: Ineuron
```

In [6]:
```
5. In the regex which created from the r'(\d\d\d)-(\d\d\d-\d\d\d\d)',
what does group zero cover? Group 2? Group 1?
Ans: In the Regex r'(\d\d\d)-(\d\d\d-\d\d\d\d)' the zero group covers the enti
    the first group cover (\d\d\d) and the second group cover (\d\d\d-\d\d\d\d

# Example Program

import re
phoneNumRegex = re.compile(r'(\d\d\d)-(\d\d\d-\d\d\d\d)')
mo = phoneNumRegex.search('My number is 415-555-4242.')
print(mo.groups()) # Prints all groups in a tuple format
print(mo.group()) # Always returns the fully matched string
print(mo.group(1)) # Returns the first group
print(mo.group(2)) # Returns the second group
```

```
('415', '555-4242')
415-555-4242
415
555-4242
```

In [7]:
```
6. In standard expression syntax, parentheses and intervals have distinct mean
  How can you tell a regex that you want it to fit real parentheses and period
Answer : The \. \( and \) escape characters in the raw string passed to re.com
    will match actual parenthesis characters
# Example Program
import re
phoneNumRegex = re.compile(r'(\(\d\d\d\)) (\d\d\d-\d\d\d\d)')
mo = phoneNumRegex.search('My phone number is (415) 555-4242.')
```

```
(415) 555-4242
```

In [8]:
```
7. The findall() method returns a string list or a list of string tuples.
 What causes it to return one of the two options?
Ans: If the regex pattern has no groups, a list of strings matched is returned
    if the regex pattern has groups, a list of tuple of strings is returned.

# Example Program
import re
phoneNumRegex = re.compile(r'(\(\d\d\d\)) (\d\d\d-\d\d\d\d)')
mo = phoneNumRegex.findall('My phone number is (415) 555-4242.')
print(mo)

# Example Program
import re
phoneNumRegex = re.compile(r'\d{3}-\d{3}-\d{4}')
mo = phoneNumRegex.findall('My number is 415-555-4242.')
```

```
[('(415)', '555-4242')]
['415-555-4242']
```

In [ ]:
```
8. In standard expressions, what does the | character mean?
```

In [9]:
```
9. In regular expressions, what does the ? character stand for?
Ans: In regular Expressions, ? characters represents zero or
    one match of the preceeding group

# Example Program
import re
match_1 = re.search("Bat(wo)?man","Batman returns")
print(match_1)
match_2 = re.search("Bat(wo)?man","Batwoman returns")
```

```
<re.Match object; span=(0, 6), match='Batman'>
<re.Match object; span=(0, 8), match='Batwoman'>
```

In [10]:
```
10.In regular expressions, what is the difference between the + and * characte
Ans: In Regular Expressions, * Represents Zero ore more occurances of the prec
    whereas + represents one or more occurances of the preceeding group.
```

```
# for Example
import re
match_1 = re.search("Bat(wo)*man","Batman returns")
print(match_1)
match_2 = re.search("Bat(wo)+man","Batman returns")
```

```
<re.Match object; span=(0, 6), match='Batman'>
None
```

In [11]:
```
11. What is the difference between {4} and {4,5} in regular expression?
Ans: {4} means that its preceeding group should repeat 4 times. where as {4,5}
    that its preceeding group should repeat mininum 4 times and maximum 5 time

# for example
import re
haRegex = re.compile(r'(Ha){3}')
mo1 = haRegex.search('HaHaHa')
mo2 = haRegex.search('Ha')
print(mo1.group())
```

```
HaHaHa
None
```

In [ ]:
```
12. What do you mean by the \d, \w, and \s shorthand character classes signify
Ans: \d, \w and \s are special sequences in regular expresssions in python:

1) \w - Matches a word character equivalent to [a-zA-Z0-9_]
2) \d - Matches digit character equivalent to [0-9]
```

In [ ]:
```
13. What do means by \D, \W, and \S shorthand character classes signify in reg
Ans: \D, \W and \S are special sequences in regular expresssions in python:

1) \W - Matches any non-alphanumeric character equivalent to [^a-zA-Z0-9_]
2) \D - Matches any non-digit character, this is equivalent to the set class [
```

In [ ]:
```
14. What is the difference between .*? and .*?
Ans: .* is a Greedy mode, which returns the longest string that meets the cond
    Whereas .*? is a non greedy mode which returns the shortest string that me
```

In [ ]:
```
15. What is the syntax for matching both numbers and lowercase letters with a
Ans: The Synatax is Either [a-z0-9] or [0-9a-z]
```

In [ ]:
```
16. What is the procedure for making a normal expression in regax case insensi
Ans: We can pass re.IGNORECASE as a flag to make a noraml expression case inse
```

In [ ]:
```
17. What does the . character normally match?
What does it match if re.DOTALL is passed as 2nd argument in re.compile()?
Ans: Dot . character matches everything in input except newline character .
     By passing re.DOTALL as a flag to re.compile(), you can make the dot char
        including the newline character.
```

In [12]:
```
18. If numReg = re.compile(r'\d+'), what will numRegex.sub('X', '11 drummers,
    five rings, 4 hen') return?
Answer: The Ouput will be 'X drummers, X pipers, five rings, X hen'
```

```
#Example
import re
numReg = re.compile(r'\d+')
numReg.sub('X', '11 drummers, 10 pipers, five rings, 4 hen')
```

Out[12]: 'X drummers, X pipers, five rings, X hen'

In [13]:
```
19. What does passing re.VERBOSE as the 2nd argument to re.compile() allow to
Ans: re.VERBOSE will allow to add whitespace and comments to string passed to

    # Without Using VERBOSE
regex_email = re.compile(r'^([a-z0-9_\.-]+)@([0-9a-z\.-]+)\.([a-z\.]{2, 6})$',

# Using VERBOSE
regex_email = re.compile(r"""
                            ^([a-z0-9_\.-]+)              # local Part like us
                            @                             # single @ sign
                            ([0-9a-z\.-]+)                # Domain name like g
                            \.                            # single Dot .
                            ([a-z]{2,6})$                 # Top level Domain
                         """,re.VERBOSE | re.IGNORECASE)
```

In [14]:
```
20. How would you write a regex that match a number with comma for every three
    It must match the given following:
'42','1,234', '6,368,745'but not the following: '12,34,567' (which has only tw
    '1234' (which lacks commas)

# Example
import re
pattern = r'^\d{1,3}(,\d{3})*$'
pagex = re.compile(pattern)
for ele in ['42','1,234', '6,368,745','12,34,567','1234']:
    print('Output:',ele, '->', pagex.search(ele))
```

```
Output: 42 -> <re.Match object; span=(0, 2), match='42'>
Output: 1,234 -> <re.Match object; span=(0, 5), match='1,234'>
Output: 6,368,745 -> <re.Match object; span=(0, 9), match='6,368,745'>
Output: 12,34,567 -> None
Output: 1234 -> None
```

In [15]:
```
21. How would you write a regex that matches the full name of someone whose la
    You can assume that the first name that comes before it will always be one
    capital letter. The regex must match the following:
'Haruto Watanabe'
'Alice Watanabe'
'RoboCop Watanabe'

but not the following:

'haruto Watanabe' (where the first name is not capitalized)
'Mr. Watanabe' (where the preceding word has a nonletter character)
'Watanabe' (which has no first name)
'Haruto watanabe' (where Watanabe is not capitalized)

Ans: pattern = r'[A-Z]{1}[a-z]*\sWatanabe'
```

```
# For Example
import re
pattern = r'[A-Z]{1}[a-z]*\sWatanabe'
namex = re.compile(pattern)
for name in ['Haruto Watanabe','Alice Watanabe','RoboCop Watanabe','haruto Wat
             'Mr. Watanabe','Watanabe','Haruto watanabe']:
    print('Output: ',name,'->',namex.search(name))
```

```
Output:  Haruto Watanabe -> <re.Match object; span=(0, 15), match='Haruto Wat
anabe'>
Output:  Alice Watanabe -> <re.Match object; span=(0, 14), match='Alice Watan
abe'>
Output:  RoboCop Watanabe -> <re.Match object; span=(4, 16), match='Cop Watan
abe'>
Output:  haruto Watanabe -> None
Output:  Mr. Watanabe -> None
Output:  Watanabe -> None
Output:  Haruto watanabe -> None
```

In [16]:

```
22. How would you write a regex that matches a sentence where the first word i
    Bob,or Carol; the second word is either eats, pets, or throws; the third w
    or baseballs; and the sentence ends with a period? This regex should be ca
    It must match the following:
'Alice eats apples.'
'Bob pets cats.'
'Carol throws baseballs.'
'Alice throws Apples.'
'BOB EATS CATS.'

but not the following:

'RoboCop eats apples.'
'ALICE THROWS FOOTBALLS.'
'Carol eats 7 cats.'

Ans: pattern = r'(Alice|Bob|Carol)\s(eats|pets|throws)\s(apples|cats|baseballs

# Example
import re
pattern = r'(Alice|Bob|Carol)\s(eats|pets|throws)\s(apples|cats|baseballs)\.'
casex = re.compile(pattern,re.IGNORECASE)
for ele in ['Alice eats apples.','Bob pets cats.','Carol throws baseballs.
','Alice throws Apples.','BOB EATS CATS.','RoboCop eats apples.'
,'ALICE THROWS FOOTBALLS.','Carol eats 7 cats.']:
```

```
Output:  Alice eats apples. -> <re.Match object; span=(0, 18), match='Alice e
ats apples.'>
```

In [ ]: