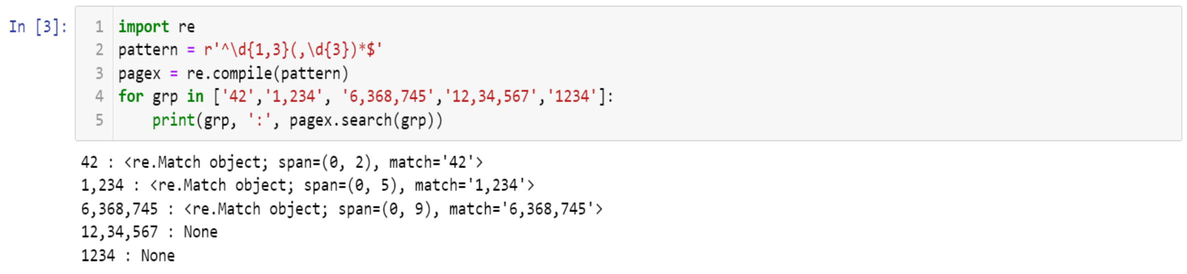
1. What is the name of the feature responsible for generating Regex objects?
Ans: re. compile() function returns Regex objects.
2. Why do raw strings often appear in Regex objects?
Ans: Raw string treats backslash (\ ) as a literal character and not as a special character.
3. What is the return value of the search() method?
Ans: search() method finds a pattern anywhere in the string and returns a match object.
4. From a Match item, how do you get the actual strings that match the pattern?
Ans: group() method returns strings of the matched text.
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: group zero covers the entire pattern: (\d\d\d)-(\d\d\d-\d\d\d\d)
Group 2 covers: \d\d\d-\d\d\d\d
Group 1 covers: \d\d\d
6. In standard expression syntax, parentheses and intervals have distinct meanings. How can you tell a regex that you want it to fit real parentheses and periods?
Ans: The \. and \ escape characters in the raw string passed to re.compile() will match actual parenthesis characters and periods.
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: findall() method returns a list of strings if the regex pattern has no groups and a list of tuples if the regex pattern has groups.
8. In standard expressions, what does the | character mean?
Ans: OR operator
9. In regular expressions, what does the ? character stand for?
Ans: ? quantifies 0 or 1 matches. Example: {1,3} : 1, 2 or 3
10.In regular expressions, what is the difference between the + and * characters?
Ans: * quantifies 0 or more. Example: {1}: Exactly 1
+ quantifies 1 or more. Example: {1,}: 1 or more
11. What is the difference between {4} and {4,5} in regular expression?
Ans: {4} means the group should repeat exactly 4 times and {4,5} means the group should repeat at least 4 times and maximum 5 times inclusively.
12. What do you mean by the \d, \w, and \s shorthand character classes signify in regular Expressions?
Ans: \d: Matches digit character equivalent to [0-9]
\w: Matches a word character equivalent to [a-z, A-Z, 0-9]
\s: Matches whitespace characters
13. What do means by \D, \W, and \S shorthand character classes signify in regular expressions?
Ans: \D: Matches any non-digit character

\W: Matches any non-word character
\S: Matches any non-whitespace character
14. What is the difference between .*? and .*?
Ans: .* is Greedy , returns the longest string that satisfies the condition. .*? is non greedy, returns the shortest string satisfying condition.
15. What is the syntax for matching both numbers and lowercase letters with a character class?
Ans: [a-z0-9] or [0-9a-z]
16. What is the procedure for making a normal expression in regex case insensitive?
Ans: pass re.IGNORECASE as a flag
17. What does the . character normally match? What does it match if re.DOTALL is passed as 2nd argument in re.compile()?
Ans: . matches everything in input except newline characters
if re.DOTALL is passed as 2nd argument in re.compile(), . matches with all characters, including the newline character.
18. If numReg = re.compile(r'\d+'), what will numRegex.sub('X', '11 drummers, 10 pipers, five rings, 4 hen') return?
Ans: 'X drummers, X pipers, five rings, X hen'
19. What does passing re.VERBOSE as the 2nd argument to re.compile() allow to do?
Ans: re.VERBOSE will allow re.compile() to add whitespace and comments to the input string.
20. How would you write a regex that match a number with comma for every three digits? It must match the given following:
'42','1,234', '6,368,745'but not the following: '12,34,567' (which has only two digits between the commas) '1234' (which lacks commas)
Ans:



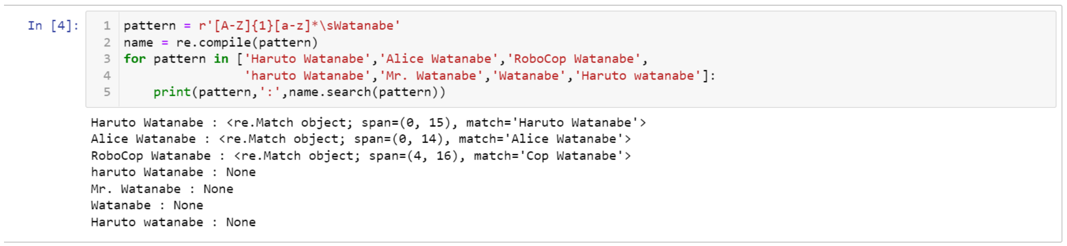
```
In [3]:
import re
pattern = r'^\d{1,3}(,\d{3})*$'
pagex = re.compile(pattern)
for grp in ['42','1,234', '6,368,745','12,34,567','1234']:
    print(grp, ':', pagex.search(grp))
```

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

21. How would you write a regex that matches the full name of someone whose last name is Watanabe? You can assume that the first name that comes before it will always be one word that begins with a 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:



```
In [4]:
pattern = r'[A-Z]{1}[a-z]*\sWatanabe'
name = re.compile(pattern)
for pattern in ['Haruto Watanabe','Alice Watanabe','RoboCop Watanabe',
                'haruto Watanabe','Mr. Watanabe','Watanabe','Haruto watanabe']:
    print(pattern,':',name.search(pattern))
```

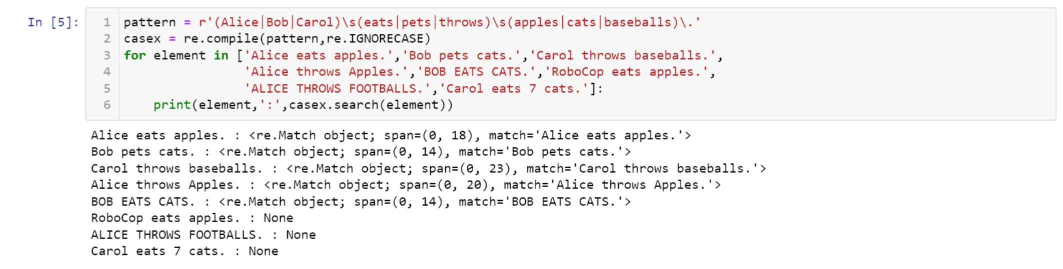
```
Haruto Watanabe : <re.Match object; span=(0, 15), match='Haruto Watanabe'>
Alice Watanabe : <re.Match object; span=(0, 14), match='Alice Watanabe'>
RoboCop Watanabe : <re.Match object; span=(4, 16), match='Cop Watanabe'>
haruto Watanabe : None
Mr. Watanabe : None
Watanabe : None
Haruto watanabe :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

22. How would you write a regex that matches a sentence where the first word is either Alice, Bob,or Carol; the second word is either eats, pets, or throws; the third word is apples, cats, or baseballs; and the sentence ends with a period? This regex should be case-insensitive. 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:



```
In [5]:
pattern = r'(Alice|Bob|Carol)\s(eats|pets|throws)\s(apples|cats|baseballs)\.'
casex = re.compile(pattern,re.IGNORECASE)
for element 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.']:
    print(element,':',casex.search(element))
```

```
Alice eats apples. : <re.Match object; span=(0, 18), match='Alice eats apples.'>
Bob pets cats. : <re.Match object; span=(0, 14), match='Bob pets cats.'>
Carol throws baseballs. : <re.Match object; span=(0, 23), match='Carol throws baseballs.'>
Alice throws Apples. : <re.Match object; span=(0, 20), match='Alice throws Apples.'>
BOB EATS CATS. : <re.Match object; span=(0, 14), match='BOB EATS CATS.'>
RoboCop eats apples. : None
ALICE THROWS FOOTBALLS. : None
Carol eats 7 cats. : None
```