**1. What is the name of the feature responsible for generating Regex objects?**

**2. Why do raw strings often appear in Regex objects?**

**3. What is the return value of the search() method?**

**4. From a Match item, how do you get the actual strings that match the pattern?**

**5. In the regex which created from the r&#39;(\d\d\d)-(\d\d\d-\d\d\d\d)&#39;, what does group zero cover?**

**Group 2? Group 1?**

**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?**

**7. The findall() method returns a string list or a list of string tuples. What causes it to return one of**

**the two options?**

**8. In standard expressions, what does the | character mean?**

**9. In regular expressions, what does the character stand for?**

**10.In regular expressions, what is the difference between the + and \* characters?**

**11. What is the difference between {4} and {4,5} in regular expression?**

**12. What do you mean by the \d, \w, and \s shorthand character classes signify in regular**

**expressions?**

**13. What do means by \D, \W, and \S shorthand character classes signify in regular expressions?**

**14. What is the difference between .\*? and .\*?**

**15. What is the syntax for matching both numbers and lowercase letters with a character class?**

**16. What is the procedure for making a normal expression in regax case insensitive?**

**17. What does the . character normally match? What does it match if re.DOTALL is passed as 2nd**

**argument in re.compile()?**

**18. If numReg = re.compile(r&#39;\d+&#39;), what will numRegex.sub(&#39;X&#39;, &#39;11 drummers, 10 pipers, five rings, 4**

**hen&#39;) return?**

**19. What does passing re.VERBOSE as the 2nd argument to re.compile() allow to do?**

**20. How would you write a regex that match a number with comma for every three digits? It must**

**match the given following:**

**&#39;42&#39;**

**&#39;1,234&#39;**

**&#39;6,368,745&#39;**

**but not the following:**

**&#39;12,34,567&#39; (which has only two digits between the commas)**

**&#39;1234&#39; (which lacks commas)**

**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:**

**&#39;Haruto Watanabe&#39;**

**&#39;Alice Watanabe&#39;**

**&#39;RoboCop Watanabe&#39;**

**but not the following:**

**&#39;haruto Watanabe&#39; (where the first name is not capitalized)**

**&#39;Mr. Watanabe&#39; (where the preceding word has a nonletter character)**

**&#39;Watanabe&#39; (which has no first name)**

**&#39;Haruto watanabe&#39; (where Watanabe is not capitalized)**

**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:**

**&#39;Alice eats apples.&#39;**

**&#39;Bob pets cats.&#39;**

**&#39;Carol throws baseballs.&#39;**

**&#39;Alice throws Apples.&#39;**

**&#39;BOB EATS CATS.&#39;**

**but not the following:**

**&#39;RoboCop eats apples.&#39;**

**&#39;ALICE THROWS FOOTBALLS.&#39;**

**&#39;Carol eats 7 cats.&#39;**

**SOLUTIONS**

1. The feature responsible for generating Regex objects is the **re** module in Python.
2. Raw strings are often used in Regex objects to avoid having to escape any special characters in the regular expression. Raw strings are denoted by placing an **r** before the string, e.g. **r'expression'**.
3. The return value of the **search()** method is a **Match** object if a match is found, and **None** if no match is found.
4. To get the actual strings that match the pattern from a **Match** object, you can use the **group()** method. You can also use the **group(n)** method to get the n-th match, where n is the group number.
5. Group 0 covers the entire expression matched by the regular expression. Group 1 covers the first set of parentheses in the expression, and group 2 covers the second set of parentheses in the expression.
6. To tell a regex that you want it to match real parentheses and periods, you can escape them with a backslash (e.g. **\(** and **\)** for parentheses, and **\.** for periods).
7. The **findall()** method returns a string list if only one set of parentheses is used in the expression, and a list of string tuples if multiple sets of parentheses are used in the expression.
8. In standard expressions, the **|** character means "or". For example, **A|B** would match either the letter **A** or the letter **B**.
9. The **.** character in regular expressions matches any character except a newline character.
10. The **+** character in regular expressions means "one or more occurrences of the preceding character". The **\*** character means "zero or more occurrences of the preceding character".
11. The syntax **{4}** in regular expressions means "exactly 4 occurrences of the preceding character", while **{4,5}** means "between 4 and 5 occurrences of the preceding character".
12. The shorthand character classes **\d**, **\w**, and **\s** signify "digit", "word character", and "whitespace character", respectively.
13. The shorthand character classes **\D**, **\W**, and **\S** signify "non-digit", "non-word character", and "non-whitespace character", respectively.
14. The difference between **.\*?** and **.\*** is that **.\*?** is a non-greedy version of **.\***. The non-greedy version will match as few characters as possible, while the greedy version will match as many characters as possible.
15. The syntax for matching both numbers and lowercase letters with a character class is **[a-z0-9]**.
16. To make a regular expression case-insensitive, you can include the **re.IGNORECASE** flag as a second argument when compiling the expression using the **re.compile()** method.
17. The **.** character normally matches any character except a newline character. If **re.DOTALL** is passed as the second argument in **re.compile()**, the **.** character will match any character including newline characters.
18. The expression **numRegex.sub('X', '11 drummers, 10 pipers, five rings, 4 hen')** would return the string **'X drummers, X pipers, five rings, X hen'**.
19. The **re.VERBOSE** flag allows to include whitespace and comments within the regular expression pattern. This makes the pattern more readable and easier to maintain. When the **re.VERBOSE** flag is passed as the second argument to **re.compile()**, whitespace within the pattern is ignored, except when escaped or included within a character class.
20. A regex to match a number with a comma for every three digits is as follows:

import re

pattern = re.compile(r'^\d{1,3}(,\d{3})\*$')

1. A regex to match the full name of someone whose last name is Watanabe is as follows:

import re

pattern = re.compile(r'^[A-Z][a-z]+\sWatanabe$')

import re

1. A regex to match 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 is as follows:

S

import re

pattern = re.compile(r'^(Alice|Bob|Carol)\s(eats|pets|throws)\s(apples|cats|baseballs)\.', re.IGNORECASE)