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

import re

re.compile("string")

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

Raw strings are used so that backslashes do not have to be escaped

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

The search() method searches a string for a specified value, and returns the position of the match. The search value can be string or a regular expression. This method returns -1 if no match is found

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

phoneNumRegex **=** re**.**compile(r'\d\d\d-\d\d\d-\d\d\d\d')

mo **=** phoneNumRegex**.**search('My number is 345-444-4542.')

print('Phone number is: ' **+** mo**.**group())

**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?**

phoneNumRegex **=** re**.**compile(r'(\d\d\d)-(\d\d\d-\d\d\d\d)')

mo **=** phoneNumRegex**.**search('My number is 345-444-4542.')

**mo.group(1)**

**output: - '345'**

**mo.group(2)**

**output: - '444-4542'**

**mo.group(0)**

**output: - '345-444-4542'**

**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?**

Periods and parentheses can be escaped with a backslash: \., \(, and \).

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

If the regex has no groups, a list of strings is returned. If the regex has groups, a list of tuples of strings is returned.

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

The | character is called a pipe. You can use it anywhere you want to match one of many expressions. For example, the regular expression r'Cricket|Soccer Sport' will match either 'Cricket' or 'Soccer Sport'.

When both Cricket and Soccer Sport occur in the searched string, the first occurrence of matching text will be returned as the Match object.

The | character signifies matching “either, or” between two groups

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

In regular Expressions, ? characters represents zero or one match of the preceding group.

**import** re

match\_1 **=** re**.**search("Bat(wo)?man","Batman returns")

print(match\_1)

match\_2 **=** re**.**search("Bat(wo)?man","Batwoman returns")

print(match\_2)

Output:

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

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

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

The + matches one or more. The \* matches zero or more

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

The {4} matches exactly 4 instances of the preceding group.

The {4,5} matches between 4 and 5 instances

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

The \d, stands for single digit, Any numeric digit from 0 to 9 \w, stands for single word, Any letter, numeric digit, or the underscore character. (Think of this as matching “word” characters.) \s stands for single space character, Any space, tab, or newline character.

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

 \D - > Any character that is not a numeric digit from 0 to 9.  
\W - > Any character that is not a letter, numeric digit, or the underscore character.  
\S - > Any character that is not a space, tab, or newline.

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

*The dot-star uses greedy mode: It will always try to match as much text as possible.  
.*? - To match any and all text in a non-greedy fashion, use the dot, star, and question mark (.\*?). Like with braces, the question mark tells Python to match in a non-greedy way.

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

Either [0-9a-z] or [a-z0-9]

reg1 **=** re**.**compile(r'[0-9a-z]')

reg2 **=** re**.**compile(r'[a-z0-9]')

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

Passing re.I or re.IGNORECASE as the second argument to re.compile() will make the matching case insensitive

**17. What does the . character normally match? What does it match if re.DOTALL is passed as 2nd argument in re.compile()?**

The . character normally matches any character except the newline character. If re.DOTALL is passed as the second argument to re.compile(), then the dot will also match newline characters.

**18. If numReg = re.compile(r'\d+'), what will numRegex.sub('X', '11 drummers, 10 pipers, five rings, 4 hen') return?**

numRegex **=** re**.**compile(r'\d+')

mo **=** numRegex**.**sub('X', '11 drummers, 10 pipers, five rings, 4 hen')

mo

Output:- 'X drummers, X pipers, five rings, X hen'

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

The re.VERBOSE argument allows you to add whitespace and comments to the string passed to re.compile()

**20. How would you write a regex that match a number with comma for every three digits? It must match the given following: but not the following: '12,34,567' (which has only two digits between the commas), '1234' (which lacks commas)**

reg1 = re.compile(r'^\d{1,3}(,\d{3})\*$')

mo1 = reg1.search('42')

mo1.group()

**Output: '42'**

reg1 = re.compile(r'^\d{1,3}(,\d{3})\*$')

mo1 = reg1.search('1,234')

mo1.group()

**Output'1,234'**

reg1 = re.compile(r'^\d{1,3}(,\d{3})\*$')

mo1 = reg1.search('6,368,745')

mo1.group()

**Output: '6,368,745'**

**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: 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)**

name **=** re**.**compile(r'[A-Z][a-z]\*\sWatanabe')

reg1 **=** re**.**compile(r'^\d{1,3}(,\d{haruto Watanabe3})\*$')

mo1 **=** name**.**search('Haruto Watanabe')

mo1**.**group()

**Output: 'Haruto Watanabe'**

name **=** re**.**compile(r'[A-Z][a-z]\*\sWatanabe')

reg1 **=** re**.**compile(r'^\d{1,3}(,\d{3})\*$')

mo1 **=** name**.**search('Alice Watanabe')

mo1**.**group()

**Output: 'Alice Watanabe'**

name **=** re**.**compile(r'[A-Z][a-z]\*\sWatanabe')

reg1 **=** re**.**compile(r'^\d{1,3}(,\d{3})\*$')

mo1 **=** name**.**search('Robocop Watanabe')

mo1**.**group()

**Output: 'RoboCop Watanabe'**

**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: but not the following: 'RoboCop eats apples.', 'ALICE THROWS FOOTBALLS.', 'Carol eats 7 cats.'**

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

mo1 = name.search('Alice eats apples.')

mo1.group()

**Output: 'Alice eats apples.'**

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

mo1 = name.search('Bob pets cats.')

mo1.group()

**Output: 'Bob pets cats.'**

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

mo1 = name.search('Carol throws baseballs.')

mo1.group()

**Output: 'Carol throws baseballs.'**

name = re.compile(r'(Alice|Bob|Carol)\s(eats|pets|throws)\s(Apples|Cats|Baseballs)\.', re.IGNORECASE)

mo1 = name.search('**'**Alice throws Apples.')

mo1.group()

**Output: 'Alice throws Apples.'**

name = re.compile(r'(ALICE|BOB|CAROL)\s(EATS|PETS|THROWS)\s(APPLES|CATS|BASEBALLS)\.', re.IGNORECASE)

mo1 = name.search('BOB EATS CATS.')

mo1.group()

**Output: 'BOB EATS CATS.'**