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

Answer: **re.compile()** generates regex objects.

>>**Example**

pattern = re.compile('hello') #returns a pattern object

To find the same pattern inside another expression, we can use

result=pattern.findall('the man said hello how are you hello')

print result

OUTPUT →['hello','hello']

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

Answer: Raw string are used in regex so backslashes dont need to be escaped and can be displayed.

>> example, s = r"/n/s/tWord"

>> prog = re.compile(s)

>> prog

OUTPUT >> **re.compile(r'/n/s/tWord', re.UNICODE)**

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

Answer: search() method returns marthing values

>>example, import re

```
pattern=re.compile('jacky')

print(re.search(pattern, 'one two three jacky')
```

>> OUTPUT → <re.Match object; span=(0, 5), match='jacky'>

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

Answer: The group() method returns strings of the matched text

```
matching = re.match(r"(\d+)\.(\d+)","90.45")

matching.groups()
```

—> OUTPUT >>> ('90', '45')

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?

Answer: Group zero covers the entire pattern, group one covers the first set of parentheses and group 2 covers the second set of parentheses .

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?

Answer: We can use backslash to escape the parentheses and intervals as follows,

Period → \.,

Parentheses → \(, \)

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

Answer: If the regex doesnt contain groups, findall returns a list of strings, if the regex contains groups, then a tuple is returned by findall

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

Answer: | character means OR logical expression between two values

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

Answer: The question mark character matches zero or one occurrences of the regular expression

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

Answer: The + matches one or more occurrences of the regular expression, the * matches zero or more occurrences.

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

Answer: The {4} matches three instances of the preceeding group, the {4,5} matches between 4 and 5 instances of the preceding group.

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

Answer: \d signifies single digit, \w signifies single word and \s signifies space character.

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

Answer: \D signifies a single character that is nota digit, \W signifies a single word and \S signifies a single white space.

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

Answer: **The .** → The Dot is a wildcard, that will match anything or any character.

The **.*** → Matches the previous element as many times as possible (zero or more time)

The ***?** → Matches the previous element as few times as possible(zero or more times)

The **.*?** → This can be used in example a.*?b → This matches a followed by anything ending with b

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

Answer: "^[a-z0-9]+"

Code >> **bool(re.match("^[a-z0-9]+",'PE09')) , OUTPUT — TRUE**

bool(re.match("^[a-z0-9]+",'pe09')) OUTPUT – FALSE

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

Answer: To make an expression case insensitive, we have to use the i flag. Its generally used as (?)

Example → bool(re.match("(?i)^[a-z0-9]+",'PO09')) → OUTPUT = TRUE

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

Answer: 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?

Answer: sub() function replaces the occurrences of an expression in a string with a replacement string, so here it will replace the occurrences of 11 with X

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?

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

Or in other words VERBOSE makes the regular expression more readable by allowing comments, ignoring white spaces within the pattern.

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)

Answer: **'^\d{1,3}(,\d{3})*$'**

Example >> **bool(re.match('^\d{1,3}(,\d{3})*$','6,383,745')) >>> OUTPUT → TRUE**

bool(re.match('^\d{1,3}(,\d{3})*$','1234')) >>> OUTPUT → FALSE

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)

Answer: r'[A-Za-z]*\sWatanabe'

Example >> **bool(re.match(r'[A-Za-z]*\sWatanabe','Erick Watanbe'))**

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.'

Answer: **r'(Alice|Bob|Carol)\s(eats|pets|throws)\s(apples|cats|baseballs)\.'**

Example >>>

bool(re.match(r'(Alice|Bob|Carol)\s(eats|pets|throws)\s(apples|cats|baseballs)\.','bob pets apples')) —> False

bool(re.match(r'(Alice|Bob|Carol)\s(eats|pets|throws)\s(apples|cats|baseballs)\.','Bob pets apples')) —> True