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

The feature responsible for generating Regex objects in Python is the re module.

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

Raw strings often appear in Regex objects to avoid issues with escape sequences. By prefixing a string with r, Python treats backslashes (\) as literal characters and not as escape characters.

#### Example:

python

Copy code

pattern = r'\d{3}-\d{2}-\d{4}' # Raw string to avoid escaping backslashes

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

The search() method returns a Match object if the pattern is found in the string, or None if the pattern is not found.

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

From a Match item, you can get the actual strings that match the pattern using the group() method.

#### Example:

python

Copy code

match = re.search(r'\d+', 'The number is 42')

if match:

print(match.group()) # Output: '42'

### 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?

* **Group 0**: The entire match ((\d\d\d)-(\d\d\d-\d\d\d\d))
* **Group 1**: The first set of parentheses (\d\d\d)
* **Group 2**: The second set of parentheses (\d\d\d-\d\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?

To match literal parentheses and periods in a regex, you need to escape them with a backslash (\).

#### Example:

python

Copy code

pattern = r'\(\d{3}\)\.\d{3}\.\d{4}' # Matches a pattern like (123).456.7890

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

The findall() method returns a list of strings if the regex has no capturing groups. If the regex has capturing groups, it returns a list of tuples, with each tuple containing the matched strings for each group.

#### Example:

python

Copy code

# No capturing groups

re.findall(r'\d+', '12 drummers drumming, 11 pipers piping') # ['12', '11']

# With capturing groups

re.findall(r'(\d+)', '12 drummers drumming, 11 pipers piping') # [('12',), ('11',)]

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

In standard expressions, the | character means "or". It allows you to match one of several possible patterns.

#### Example:

python

Copy code

pattern = r'cat|dog' # Matches 'cat' or 'dog'

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

In regular expressions, the . character stands for any single character except a newline.

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

* **+**: Matches one or more occurrences of the preceding element.
* **\***: Matches zero or more occurrences of the preceding element.

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

* **{4}**: Matches exactly 4 occurrences of the preceding element.
* **{4,5}**: Matches between 4 and 5 occurrences of the preceding element.

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

* **\d**: Matches any digit (equivalent to [0-9]).
* **\w**: Matches any word character (alphanumeric plus underscore, equivalent to [A-Za-z0-9\_]).
* **\s**: Matches any whitespace character (space, tab, newline, etc.).

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

* **\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 .\*?

* **.\***: Greedy match, matches as many characters as possible.
* **.\*?**: Non-greedy (lazy) match, matches as few characters as possible.

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

The syntax for matching both numbers and lowercase letters is [0-9a-z].

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

To make a regex case insensitive, you can pass the re.IGNORECASE flag to the re.compile() function.

#### Example:

python

Copy code

pattern = re.compile(r'hello', re.IGNORECASE)

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

* Normally, the . character matches any character except a newline.
* If re.DOTALL is passed, the . character matches any character including a newline.

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

It will return 'X drummers, X pipers, five rings, X hen'.

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

Passing re.VERBOSE allows you to write more readable regex by enabling you to add whitespace and comments within the pattern.

#### Example:

python

Copy code

pattern = re.compile(r'''

\d{3} # Area code

- # Separator

\d{3} # First 3 digits

- # Separator

\d{4} # Last 4 digits

''', re.VERBOSE)

### 20. How would you write a regex that matches a number with a comma for every three digits?

The regex to match a number with a comma for every three digits is:

python

Copy code

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

This matches:

* '42'
* '1,234'
* '6,368,745' but not:
* '12,34,567'
* '1234'

### 21. How would you write a regex that matches the full name of someone whose last name is Watanabe?

The regex to match the full name of someone whose last name is Watanabe is:

python

Copy code

pattern = r'[A-Z][a-z]\*\sWatanabe'

This matches:

* 'Haruto Watanabe'
* 'Alice Watanabe'
* 'RoboCop Watanabe' but not:
* 'haruto Watanabe'
* 'Mr. Watanabe'
* 'Watanabe'
* 'Haruto 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?

The regex to match such sentences case-insensitively is:

python

Copy code

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

This matches:

* 'Alice eats apples.'
* 'Bob pets cats.'
* 'Carol throws baseballs.'
* 'Alice throws Apples.'
* 'BOB EATS CATS.' but not:
* 'RoboCop eats apples.'
* 'ALICE THROWS FOOTBALLS.'
* 'Carol eats 7 cats.'