Quantifiers

In Python regular expressions, quantifiers are used to specify the number of occurrences of a character or group in a pattern. They allow you to match a certain quantity of characters, whether it's a specific number or within a range. Here are some common quantifiers and their usage:

### 1. Asterisk `*` - Zero or More:

The asterisk (*) quantifier matches zero or more occurrences of the preceding character or group.

```python
import re

text = "ab abc abbc abbbc"

# Match 'ab' followed by zero or more 'c' characters
pattern = re.compile(r'ab*c')
matches = pattern.findall(text)
print(matches)  # Output: ['ab', 'abc', 'abbc', 'abbbc']
```

### 2. Plus `+` - One or More:

The plus (+) quantifier matches one or more occurrences of the preceding character or group.

```python
# Match 'ab' followed by one or more 'c' characters
pattern = re.compile(r'ab+c')
matches = pattern.findall(text)
print(matches)  # Output: ['abc', 'abbc', 'abbbc']
```

### 3. Question Mark `?` - Zero or One:

The question mark (?) quantifier matches zero or one occurrence of the preceding character or group.

```python
# Match 'ab' followed by zero or one 'c' characters
pattern = re.compile(r'ab?c')
matches = pattern.findall(text)
print(matches)  # Output: ['abc', 'ac']
```

### 4. Curly Braces `{}` - Specific Number of Occurrences:

Curly braces allow you to specify a specific number of occurrences of the preceding character or group.

```python
```

```
# Match 'ab' followed by exactly two 'c' characters
pattern = re.compile(r'ab{2}c')
matches = pattern.findall(text)
print(matches)  # Output: ['abbc']
```

### 5. Curly Braces `{min, max}` - Range of Occurrences:

You can also use curly braces to specify a range of occurrences.

```python
# Match 'ab' followed by two to three 'c' characters
pattern = re.compile(r'ab{2,3}c')
matches = pattern.findall(text)
print(matches)  # Output: ['abbc', 'abbbc']
```

### 6. Greedy and Non-Greedy Quantifiers:

By default, quantifiers are greedy, meaning they match as much as possible. Adding a question mark after a quantifier makes it non-greedy, matching as little as possible.

```python
text = "ababab"

# Greedy match - as many 'ab' pairs as possible
pattern_greedy = re.compile(r'ab*')
print(pattern_greedy.findall(text))  # Output: ['ababab']

# Non-greedy match - as few 'ab' pairs as possible
pattern_non_greedy = re.compile(r'ab*?')
print(pattern_non_greedy.findall(text))  # Output: ['a', 'a', 'a']
```

These examples cover the basic usage of quantifiers in Python regular expressions. Quantifiers provide flexibility in specifying the number of occurrences of characters or groups, making it easier to express complex patterns in your matching logic.