Character Classes

In Python regular expressions, character classes are a way to specify a set of characters that you want to match. They allow you to define a group of characters and match any one of them. Here are some common uses of character classes along with examples:

1. Square Brackets `[]`:

Square brackets define a character class. Inside the brackets, you list the characters you want to match.

```python
import re

text = "apple orange banana"

# Match any of the characters 'a', 'e', or 'o'
pattern = re.compile(r'[aeo]')
matches = pattern.findall(text)
print(matches)  # Output: ['a', 'o', 'e', 'o', 'a', 'a', 'a']
```

2. Character Ranges:

You can specify a range of characters using a hyphen inside square brackets.

```python
# Match any lowercase letter
pattern = re.compile(r'[a-z]')
matches = pattern.findall(text)
print(matches)  # Output: ['a', 'p', 'p', 'l', 'e', 'o', 'r', 'a', 'n', 'g', 'e', 'b', 'a', 'n', 'a', 'n', 'a']
```

3. Negation `^`:

Placing a `^` at the beginning of a character class negates it, meaning it matches any character not in the specified class.

```python
# Match any character that is not a vowel
pattern = re.compile(r'[^aeiou]')
matches = pattern.findall(text)
print(matches)  # Output: ['p', 'p', 'l', ' ', 'r', 'n', 'g', ' ', 'b', 'n', 'n']
```

4. Predefined Character Classes:

There are some shorthand notations for common character classes:

- `\d`: Matches any digit (equivalent to `[0-9]`).
- `\D`: Matches any non-digit.
- `\w`: Matches any word character (alphanumeric + underscore).
- `\W`: Matches any non-word character.
- `\s`: Matches any whitespace character.
- `\S`: Matches any non-whitespace character.

```python
text = "abc 123 !@#"

# Match digits
pattern = re.compile(r'\d')
print(pattern.findall(text))  # Output: ['1', '2', '3']

# Match non-digits
pattern = re.compile(r'\D')
print(pattern.findall(text))  # Output: ['a', 'b', 'c', ' ', '!', '@', '#']

# Match word characters
pattern = re.compile(r'\w')
print(pattern.findall(text))  # Output: ['a', 'b', 'c', '1', '2', '3']

# Match non-word characters
pattern = re.compile(r'\W')
print(pattern.findall(text))  # Output: [' ', '!', '@', '#']

# Match whitespace characters
pattern = re.compile(r'\s')
print(pattern.findall(text))  # Output: [' ', ' ']

# Match non-whitespace characters
pattern = re.compile(r'\S')
print(pattern.findall(text))  # Output: ['a', 'b', 'c', '1', '2', '3', '!', '@', '#']
```

These examples demonstrate the basic usage of character classes in Python regular expressions. They are powerful tools for pattern matching and allow for flexible and concise specification of character sets.