Match Objects

In Python regular expressions, when a match is found, it returns a match object that provides information about the match. Match objects have methods and attributes that allow you to retrieve details about the match. Let's go through some examples to illustrate the use of match objects:

1. Using `re.match()`:

The `re.match()` function attempts to match the pattern only at the beginning of the string. If successful, it returns a match object; otherwise, it returns `None`.

```python
import re

text = "Hello, World!"
pattern = re.compile(r'Hello')

# Try to match the pattern at the beginning of the string
match_object = pattern.match(text)

if match_object:
    print("Match found:", match_object.group())
else:
    print("No match")
```

In this example, the pattern `Hello` matches at the beginning of the string, so a match object is returned.

2. Using `re.search()`:

The `re.search()` function searches the entire string for the pattern and returns the first match object found.

```python
# Search for the pattern anywhere in the string
search_object = pattern.search(text)

if search_object:
    print("Search found:", search_object.group())
else:
    print("No match")
```

Here, the pattern `Hello` is found later in the string, so a match object is returned.

3. Accessing Match Object Attributes:

Match objects have several attributes, such as `group()`, `start()`, and `end()`, that provide information about the matched substring.

```python
text = "The price is $20.50"
pattern = re.compile(r'\$\d+\.\d+')

match_object = pattern.search(text)

if match_object:
    print("Matched text:", match_object.group())
    print("Start index:", match_object.start())
    print("End index:", match_object.end())
    print("Match span:", match_object.span())
else:
    print("No match")
```

In this example, the pattern `\$\d+\.\d+` matches the currency value in the string, and the match object provides information about the match, including the matched text, start and end indices, and the span of the match.

4. Using Grouping:

You can use grouping with parentheses to capture specific parts of the match.

```python
text = "Date: 2022-01-01"
pattern = re.compile(r'Date: (\d{4}-\d{2}-\d{2})')

match_object = pattern.search(text)

if match_object:
    print("Matched text:", match_object.group())
    print("Captured date:", match_object.group(1))
else:
    print("No match")
```

Here, the pattern captures the date portion using parentheses, and the match object allows you to access the entire match and the specific captured group.

Understanding match objects is crucial when working with regular expressions in Python, as they provide detailed information about the matches, enabling you to extract and manipulate the matched substrings as needed.