simple_metacharacters

March 18, 2023

1 Simple MetaCharacters

As we indicated in a previous lesson, regular expressions use metacharacters to give special instructions. Here again is a complete list of all the metacharacters used in regular expressions:

```
. ^ $ * + ? { } [ ] \ | ( )
```

We already learned how to use one of these metacharacters, the backslash (\), to create special sequences. In the following lessons we will learn how to use the remaining metacharacters to create more complicated regular expressions.

In this notebook, we will take a look at the following metacharacters:

```
. ^ $
```

Let's start by looking at the dot (`.`) metacharacter.

1.0.1 The Dot (`.`)

As we saw in a previous lesson, the dot (`.`) matches any character except for newline (`\n`) characters. In the code below, we will use `.` as our regular expression to find all the characters in our multi-line `sample_text` string:

```
In [1]: # Import re module
        import re

        # Sample text
        sample_text = '''
        \tAlice lives in:\f
        1230 First St.\r
        Ocean City, MD 156789.\v
        '''

        # Create a regular expression object with the regular expression '.'
        regex = re.compile(r'.')

        # Search the sample_text for the regular expression
        matches = regex.finditer(sample_text)
        # Print all the matches
        for match in matches:
            print(match)

<_sre.SRE_Match object; span=(1, 2), match='\t'>
<_sre.SRE_Match object; span=(2, 3), match='A'>
<_sre.SRE_Match object; span=(3, 4), match='l'>
<_sre.SRE_Match object; span=(4, 5), match='i'>
<_sre.SRE_Match object; span=(5, 6), match='c'>
<_sre.SRE_Match object; span=(6, 7), match='e'>
<_sre.SRE_Match object; span=(7, 8), match=' '>
<_sre.SRE_Match object; span=(8, 9), match='l'>
<_sre.SRE_Match object; span=(9, 10), match='i'>
<_sre.SRE_Match object; span=(10, 11), match='v'>
<_sre.SRE_Match object; span=(11, 12), match='e'>
<_sre.SRE_Match object; span=(12, 13), match='s'>
<_sre.SRE_Match object; span=(13, 14), match=' '>
<_sre.SRE_Match object; span=(14, 15), match='i'>
<_sre.SRE_Match object; span=(15, 16), match='n'>
<_sre.SRE_Match object; span=(16, 17), match=':'>
<_sre.SRE_Match object; span=(17, 18), match='\x0c'>
<_sre.SRE_Match object; span=(19, 20), match='1'>
<_sre.SRE_Match object; span=(20, 21), match='2'>
<_sre.SRE_Match object; span=(21, 22), match='3'>
<_sre.SRE_Match object; span=(22, 23), match='0'>
<_sre.SRE_Match object; span=(23, 24), match=' '>
<_sre.SRE_Match object; span=(24, 25), match='F'>
<_sre.SRE_Match object; span=(25, 26), match='i'>
<_sre.SRE_Match object; span=(26, 27), match='r'>
<_sre.SRE_Match object; span=(27, 28), match='s'>
<_sre.SRE_Match object; span=(28, 29), match='t'>
<_sre.SRE_Match object; span=(29, 30), match=' '>
<_sre.SRE_Match object; span=(30, 31), match='S'>
<_sre.SRE_Match object; span=(31, 32), match='t'>
<_sre.SRE_Match object; span=(32, 33), match='.'>
<_sre.SRE_Match object; span=(33, 34), match='\r'>
<_sre.SRE_Match object; span=(35, 36), match='O'>
<_sre.SRE_Match object; span=(36, 37), match='c'>
<_sre.SRE_Match object; span=(37, 38), match='e'>
<_sre.SRE_Match object; span=(38, 39), match='a'>
<_sre.SRE_Match object; span=(39, 40), match='n'>
<_sre.SRE_Match object; span=(40, 41), match=' '>
<_sre.SRE_Match object; span=(41, 42), match='C'>
<_sre.SRE_Match object; span=(42, 43), match='i'>
<_sre.SRE_Match object; span=(43, 44), match='t'>
<_sre.SRE_Match object; span=(44, 45), match='y'>
<_sre.SRE_Match object; span=(45, 46), match=','>
<_sre.SRE_Match object; span=(46, 47), match=' '>
```

```
<_sre.SRE_Match object; span=(47, 48), match='M'>
<_sre.SRE_Match object; span=(48, 49), match='D'>
<_sre.SRE_Match object; span=(49, 50), match=' '>
<_sre.SRE_Match object; span=(50, 51), match='1'>
<_sre.SRE_Match object; span=(51, 52), match='5'>
<_sre.SRE_Match object; span=(52, 53), match='6'>
<_sre.SRE_Match object; span=(53, 54), match='7'>
<_sre.SRE_Match object; span=(54, 55), match='8'>
<_sre.SRE_Match object; span=(55, 56), match='9'>
<_sre.SRE_Match object; span=(56, 57), match='.'>
<_sre.SRE_Match object; span=(57, 58), match='\x0b'>
```

As we can see, we were able to match all the characters in our `sample_text` string, except for newline characters.

1.0.2 The Caret (ˆ)

The caret (ˆ) is used to match a sequence of characters when they appear at the beginning of a string. Let's take a look at an example.

In the code below, our `sample_text` string has the word `this` written twice:

```
this watch belongs in this box.
```

As we can see, the first instance of the word `this` occurs at the beginning of the string; while the second instance of the word `this` occurs towards the end of the string.

If we use `this` as our regular expression, we will match both instances of the word as shown in the code below:

```
In [2]: # Import re module
        import re

        # Sample text
        sample_text = 'this watch belongs in this box.'

        # Create a regular expression object with the regular expression 'this'
        regex = re.compile(r'this')

        # Search the sample_text for the regular expression
        matches = regex.finditer(sample_text)

        # Print all the matches
        for match in matches:
            print(match)
```

```
<_sre.SRE_Match object; span=(0, 4), match='this'>
<_sre.SRE_Match object; span=(22, 26), match='this'>
```

We can clearly see that we get two matches that correspond to both instances of the word `this` in our `sample_text` string.

Now, let's use the caret to only find the word `this` that appears at the beginning of the string. We can do this by adding the caret (^) before the word `this` in our regular expression as shown below:

```
In [3]: # Import re module
        import re

        # Sample text
        sample_text = 'this watch belongs in this box.'

        # Create a regular expression object with the regular expression '^this'
        regex = re.compile(r'^this')

        # Search the sample_text for the regular expression
        matches = regex.finditer(sample_text)

        # Print all the matches
        for match in matches:
            print(match)
```

```
<_sre.SRE_Match object; span=(0, 4), match='this'>
```

We can see that now, we only get one match, corresponding to the word `this` that appears at the beginning of the string. It didn't match the second instance of word `this` because it wasn't at the beginning of our `sample_text` string.

1.0.3 The Dollar Sign ($)

The dollar sign ($) is used to match a sequence of characters when they appear at the end of a string. Let's take a look at an example.

In the code below, our `sample_text` string has the word `watch` written twice:

```
this watch is better than this watch
```

As we can see, the first instance of the word `watch` occurs towards the beginning of the string; while the second instance of the word `watch` occurs at the end of the string.

If we use `watch` as our regular expression, we will match both instances of the word as shown in the code below:

```
In [4]: # Import re module
        import re

        # Sample text
        sample_text = 'this watch is better than this watch'

        # Create a regular expression object with the regular expression 'watch'
        regex = re.compile(r'watch')

        # Search the sample_text for the regular expression
        matches = regex.finditer(sample_text)

        # Print all the matches
        for match in matches:
            print(match)
```

```
<_sre.SRE_Match object; span=(5, 10), match='watch'>
<_sre.SRE_Match object; span=(31, 36), match='watch'>
```

We can clearly see that we get two matches that correspond to both instances of the word `watch` in our `sample_text` string.

Now, let's use the dollar sign to only find the word `watch` that appears at the end of the string. We can do this by adding the dollar sign ($) after the word `watch` in our regular expression as shown below:

```
In [5]: # Import re module
        import re

        # Sample text
        sample_text = 'this watch is better than this watch'

        # Create a regular expression object with the regular expression 'watch$'
        regex = re.compile(r'watch$')

        # Search the sample_text for the regular expression
        matches = regex.finditer(sample_text)

        # Print all the matches
        for match in matches:
            print(match)
```

```
<_sre.SRE_Match object; span=(31, 36), match='watch'>
```

We can see that now, we only get one match, corresponding to the word `watch` that appears at the end of the string. It didn't match the first instance of word `watch` because it wasn't at the end of our `sample_text` string.

```
In [ ]:
```