simple_patterns_solution

March 18, 2023

1 TODO: Find IP Addresses

In the cell below, our `sample_text` string contains three IP addresses. Write a single regular expression that can match any IP address and save the regular expression object in a variable called `regex`. Then use the `.finditer()` method to search the `sample_text` string for the given regular expression. Finally, write a loop to print all the `matches` found by the `.finditer()` method.

HINT : Use the special sequence `\d` and take advantage that all IP addresses have the same pattern.

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

        # Sample text
        sample_text = 'Here are three IP address: 123.456.789.123, 999.888.777.666, 111.222.333.

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

        # 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=(27, 42), match='123.456.789.123'>
<_sre.SRE_Match object; span=(44, 59), match='999.888.777.666'>
<_sre.SRE_Match object; span=(61, 76), match='111.222.333.444'>
```

If you wrote your regex correctly you should see three matches above corresponding to the three IP addresses in our `sample_text` string.

2 TODO: Print The Numbers Between Whitespace Characters

In the cell below, our `sample_text` consists of a multi-line string with numbers in between whitespace characters:

```
123 45  7895
1   222 33
```

Notice that not all the numbers have the same number of digits. For example, the first number (`123`) has three digits, but the second number (`45`) only has two digits.

Notice that not all the numbers have the same number of digits. For example, the first number (`123`) has three digits, but the second number (`45`) only has two digits.

Write a single regular expression that finds the tabs (`\t`) and the newlines (`\n`) in this multi-line string and save the regular expression object in a variable called `regex`. Then use the `.finditer()` method to search the `sample_text` string for the given regular expression. Then, write a loop that uses the span information from each `match` to only print the numbers found in the original multi-line string. Your code should work in the general case where the numbers can have any number of digits. For example, if the numbers in the string were to change your code should still be able to find them and print them. Finally, in this exercise you cannot use `\d` in your regular expression.

HINT : Notice that there are no whites paces in the multiline string. Use the `\s` sequence to find the tabs and newlines. Then notice that you can use the span's `end` and `start` index from consecutive matches to figure out the number of digits of each number. Use these indices to print the numbers found in the original multi-line string. You can use the `match.span()` method we saw before to find the `start` and `end` indices of each `match`. Alternatively, you can also use the `.start()` and `.end()` methods to extract the `start` and `end` indices of each match. The `match.start()` is equivalent to `match.span()[0]` and `match.end()` is equivalent to `match.span()[1]`.

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

        # Sample text
        sample_text = '''
        123\t45\t7895
        1\t222\t33
        '''

        # Print sample_text
        print('Sample Text:\n', sample_text)

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

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

        # Set counter
        counter = 0

        # Write a loop to print all the numbers found in the original string
        for match in matches:
            if counter != 0:
                start_idx = match.start()
                print('\nNumbers from the original text:', sample_text[end_idx:start_idx])
```

```
            end_idx = match.end()
            counter += 1

Sample Text:

123        45        7895
1        222        33


Numbers from the original text: 123

Numbers from the original text: 45

Numbers from the original text: 7895

Numbers from the original text: 1

Numbers from the original text: 222

Numbers from the original text: 33
```

3 TODO: Find emails

In the cell below, our `sample_text` consists of a multi-line string that contains three email addresses:

```
j.s@email.com
a.w@email.com
m.j@email.com
```

Notice, that all three email address have the same pattern, namely, the first name initial, followed by a dot (`.`), followed by the last name initial, and ending in `@email.com`.

Take advantage of the fact that all three email addresses have the same pattern to write a single regular expression that can find all three email addresses in our `sample_text` string. As usual, save the regular expression object in a variable called `regex`. Then use the `.finditer()` method to search the `sample_text` string for the given regular expression. Finally, write a loop to print all the `matches` found by the `.finditer()` method.

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

        # Sample text
        sample_text = '''
        John Sanders: j.s@email.com
        Alice Walters: a.w@email.com
        Mary Jones: m.j@email.com
        '''
```

```
        # Print sample_text
        print('Sample Text:\n', sample_text)

        # Create a regular expression object with the regular expression
        regex = re.compile(r'\w\.\w@email.com')

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

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

```
Sample Text:

John Sanders: j.s@email.com
Alice Walters: a.w@email.com
Mary Jones: m.j@email.com

<_sre.SRE_Match object; span=(15, 28), match='j.s@email.com'>
<_sre.SRE_Match object; span=(44, 57), match='a.w@email.com'>
<_sre.SRE_Match object; span=(70, 83), match='m.j@email.com'>
```

If you wrote your regex correctly you should see three matches above corresponding to the three email addresses found in our `sample_text` string.

```
In [ ]:
```