Working With Strings In Pandas: Takeaways

by Dataquest Labs, Inc. - All rights reserved © 2020

Syntax

REGULAR EXPRESSIONS

- To match multiple characters, specify the characters between "[ ]":

```
pattern = r"[Nn]ational accounts"
```

  - This expression would match "national accounts" and "National accounts".

- To match a range of characters or numbers, use:

```
pattern = r"[0-9]"
```

  - This expression would match any number between 0 and 9.

- To create a capturing group, or indicate that only the character pattern matched should be extracted, specify the characters between "( )":

```
pattern = r"([1-2][0-9][0-9][0-9])"
```

  - This expression would match years.

- To repeat characters, use "{ }". To repeat the pattern "[0-9]" three times:

```
pattern = r"([1-2][0-9]{3})"
```

  - This expression would also match years.

- To name a capturing group, use:

```
pattern = r"(?P<Years>[1-2][0-9]{3})"
```

  - This expression would match years and name the capturing group "Years".

VECTORIZED STRING METHODS

- Find specific strings or substrings in a column:

```
df[col_name].str.contains(pattern)
```

- Extract specific strings or substrings in a column:

```
df[col_name].str.extract(pattern)
```

- Extract more than one group of patterns from a column:

```
df[col_name].str.extractall(pattern)
```

- Replace a regex or string in a column with another string:

```
df[col_name].str.replace(pattern, replacement_string)
```

Concepts

- Pandas has built in a number of vectorized methods that perform the same operations for strings in Series as Python string methods.
- A regular expression is a sequence of characters that describes a search pattern. In pandas, regular expressions is integrated with vectorized string methods to make finding and extracting patterns of characters easier.

Resources

- Working with Text Data
- Regular Expressions

Takeaways by Dataquest Labs, Inc. - All rights reserved © 2020