Data Analysis with Python

Pandas Part-2

To do: Go through the activity to the end

Regular Expressions (REGEX)

- A regular expression (shortened as regex) is a sequence of characters that specifies a search pattern in text. Usually such patterns are used by string-searching algorithms for "find" or "find and replace" operations on strings or for input validation.

Regular expressions use the backslash character (`'\'`) to indicate special forms or to allow special characters to be used without invoking their special meaning. Python's raw string notation for regular expression patterns fixes the problem of using special characters. `r"\n"` is a two-character string containing `'\'` and `'n'`, while `"\n"` is a one-character string containing a newline.

Common Patterns

`\d` Any numeric digit from `0` to `9`.

`\D` Matches any character which is not a decimal digit. This is the opposite of `\d`.

`\w` Any letter, numeric digit, or the underscore character. (Think of this as matching "word" characters.)

`\W` Any character that is not a letter, numeric digit, or the underscore character.

`\s` Any space, tab, or newline character. (Think of this as matching white-space characters.)

`\S` Any character that is not a space, tab, or newline.

Common Metacharacters

`"[]"` A set of characters `"[a-m]"`

`"\"` Signals a special sequence (can also be used to escape special characters)

`"."` Any character (except newline character)

`"^"` Starts with `"^hello"`

`"$"` Ends with `"world$"`

`"*"` Zero or more occurrences

"+" One or more occurrences

"{}" Exactly the specified number of occurrences

"|" Either or "falls|stays"

"()" Capture and group

Raw String ("r / R")

- Python raw string is created by prefixing a string literal with 'r' or 'R'.
- Python raw string treats backslash (\) as a literal character. This is useful when we want to have a string that contains backslash and don't want it to be treated as an escape character

print("Backslash: \\") ------ Backslash: \
print(r"Backslash: \\") ----- Backslash: \\

Common Python RegEx Functions

re.search(pattern, string, flags=0)

Scan through string looking for a match to the pattern, returning a Match object, or None if no match was found

text = "A78L41K"
num = re.search("78", text)
num.group() ---->> '78'

re.match(pattern, string, flags=0)

Try to apply the pattern at the start of the string, returning a Match object, or None if no match was found.

If you want to locate a match anywhere in string, use search() instead of match()

text = "A78L41K"
alpha = re.match("\D\d\d", text)
alpha.group() ----->> 'A78'

re.fullmatch(pattern, string, flags=0)

Try to apply the pattern to all of the string

alpnum = re.fullmatch("\D\d+\D\d+\D", text)
alpnum.group() --->> 'A78L41K'

re.findall(pattern, string, flags=0)

Return a list of all non-overlapping matches in the string.

text = "O 1, t 10, o 100. 100000"

re.findall("\d{1}", text) ---------->> ['1', '1', '0', '1', '0', '0', '1', '0', '0', '0', '0', '0']

`re.sub(pattern, repl, string, count=0, flags=0)`

Return the string obtained by replacing the leftmost non-overlapping occurrences of the pattern in string by the replacement repl. repl can be either a string or a callable; if a string, backslash escapes in it are processed. If it is a callable, it's passed the Match object and must return a replacement string to be used.

text = "2004-959-559 # This is Phone Number"
re.sub("\D", "", text) ----->> '2004959559'

`re.split(pattern, string, maxsplit=0, flags=0)`

Split the source string by the occurrences of the pattern, returning a list containing the resulting substrings.

text = "ab56cd78_de fg3hıi49"
re.split("\D+", text) ------>> ['', '56', '78', '3', '49']

Pandas Functions Accepting RegEx

s = pd.Series(['a3aa', 'b4aa', 'c5aa'])

print(s)

```
0    a3aa
1    b4aa
2    c5aa
```

- **count()**: Count occurrences of pattern in each string of the Series/Index

`pandas.Series.str.count(pat, flags=0)`

Count occurrences of pattern in each string of the Series/Index.

This function is used to count the number of times a particular regex pattern is repeated in each of the string elements of the Series.

```
s.str.count('a')
```

```
0    3
1    2
2    2
dtype: int64
```

- **replace()**: Replace the search string or pattern with the given value

`pandas.Series.str.replace(pat, repl, n=- 1, case=None, flags=0, regex=None)`

Replace each occurrence of pattern/regex in the Series/Index.

Equivalent to str.replace() or re.sub(), depending on the regex value.

```
s.str.replace('\d','R', regex=True)
```

```
0    aRaa
1    bRaa
2    cRaa
dtype: object
```

- **contains():** Test if pattern or regex is contained within a string of a Series or Index. Calls re.search() and returns a boolean

```
pandas.Series.str.contains(pat, case=True, flags=0, na=None, regex=True)
```

Test if pattern or regex is contained within a string of a Series or Index.

Return boolean Series or Index based on whether a given pattern or regex is contained within a string of a Series or Index.

```
s.str.contains('b')
```

```
0    False
1     True
2    False
dtype: bool
```

- **findall():** Find all occurrences of pattern or regular expression in the Series/Index. Equivalent to applying re.findall() on all elements

```
pandas.Series.str.findall(pat, flags=0)
```

Find all occurrences of pattern or regular expression in the Series/Index.

Equivalent to applying re.findall() to all the elements in the Series/Index.

```
s = pd.Series(['a3a8a', 'b4a7a', 'c5a6a'])
s
```

```
0    a3a8a
1    b4a7a
2    c5a6a
dtype: object
```

```
s.str.findall('\d')
```

```
0    [3, 8]
1    [4, 7]
2    [5, 6]
dtype: object
```

- **match():** Determine if each string matches a regular expression. Calls re.match() and returns a boolean

```
pandas.Series.str.match(pat, case=True, flags=0, na=None)
```

Determine if each string starts with a match of a regular expression.

```
s = pd.Series(['&a3a8a', 'b4a&7a', 'c5a6a'])
s
```

```
0    &a3a8a
1    b4a&7a
2     c5a6a
dtype: object
```

```
s.str.match('&')
```

```
0     True
1    False
2    False
dtype: bool
```

- **split():** Split strings around given separator/delimiter and accepts string or regular expression to split on

```
pandas.Series.str.split(pat=None, n=- 1, expand=False, *, regex=None)
```

Split strings around given separator/delimiter.

Splits the string in the Series/Index from the beginning, at the specified delimiter string.

```
s = pd.Series(['&a3a8a', 'b4a&7a', 'c5a6a'])
s
```

```
0    &a3a8a
1    b4a&7a
2     c5a6a
dtype: object
```

```
s.str.split('&')
```

```
0    [, a3a8a]
1    [b4a, 7a]
2      [c5a6a]
dtype: object
```

- **extract():** Extract capture groups in the regex pat as columns in a DataFrame and returns the captured groups

```
pandas.Series.str.extract(pat, flags=0, expand=True)
```

Extract capture groups in the regex pat as columns in a DataFrame.

For each subject string in the Series, extract groups from the first match of regular expression pat.

```
s = pd.Series(['&a3a+?8a', 'b4a&+^7a', 'c5a6a'])
s
```

```
0    &a3a+?8a
1    b4a&+^7a
2       c5a6a
dtype: object
```

```
s.str.extract('(\w+)')
```

	0
0	a3a
1	b4a
2	c5a6a

This question may not be embedded here.

This question may not be embedded here.

Previous

Next

You have completed 77% of the lesson

77%

Jump to...

Reset user tour on this page