Python Regex

View source of this page

Importing re

- To use regexes in Python, one must first `import re`
- Without the import statement, Python will not know how to use regular expressions

```
import re
```

Compile and search

- Before diving into `re.compile()`, one needs to understand regexes and how to pattern match.
- Go to Regex Cheatsheet to learn more about how to build your pattern you are trying to match on.
- Excerpt:

Operator	Meaning
`*`	0 or more characters
`+`	1 or more characters
`?`	0 or 1 characters
`.`	one wildcard
`{n}`	exactly n characters
`\d`	0,1,2,3,4,5,6,7,8 or 9
`[a-z]`	one character between a and z
`[A-Z]`	one character between A and Z
`\`	used to escape special characters

Compile

- In Python, we must create a regex pattern object that has the pattern one is trying to match.
- For example, lets say that I am wanting to match a cell phone number with this exact pattern `123-456-7894`.
- To do so we must create a regex pattern object as seen below.

```
phone = re.compile(r'\d\d\d-\d\d\d-\d\d\d\d')
```

NOTE: the `r` means raw string which tells Python to not interpret the backslashes as escapes. This is imporant because the `re` library needs to "see" the backslashes.

Search

- Now that the regex pattern object is defined and is stored in the variable `phone`, let's search through a string using the `search()` method of regex pattern objects.
- `search()` only matches on the first instance and returns a regex match object if there is a match and `None` otherwise.
- The results of `search()` will be stored in a variable as seen below:

```
match_object = phone.search("The phone number is 123-789-7484")
```

NOTE: If one prints `match_object`, the output would be:

```
<re.Match object; span=(21, 33), match='123-789-7484'>
```

Group

- Using the `group()` method of regex match objects, one can print out the match and can also match certain elements in the match.
- To print out the match from the example above, see the print below:

```
print(match_object.group())
```

NOTE: If `search()` returned `None` there will be a runtime error when attempting to use the `group()` method on a `None` object.

Groups

- When we create our regex pattern object we can group characters together using parentheses `()`. In order to access these groupings in the match object we need to use the `groups()` method of the regex match object. The `groups()` method returns a tuple of all groups within the match object.
- For example, lets say someone is looking for just the area code of the phone number. We could put () around the first three digits and we could print that out.

```python
import re
phone = re.compile(r'(\d\d\d)-\d\d\d-\d\d\d\d')
match_object = phone.search("The phone numbber is 123-789-7484")
print(match_object.groups())
```

Findall

- Using the `findall()` method of regex pattern objects, one can get all the matches returned as a list.
- An example is provided below:

```python
import re
phone = re.compile(r'\d\d\d-\d\d\d-\d\d\d\d')
match_object = phone.findall("The old phone number is 123-789-7484, My new number is 236-789-8794")
print(match_object)
```

```
['123-789-7484', '236-789-8794']
```

Using findall to search through a file

```python
import re
phoneNumRegex = re.compile(r'\d\d\d-\d\d\d-\d\d\d\d')
with open("find.txt") as file:
    file = file.read()
    mat = phoneNumRegex.search(file)
    fall = re.findall(phoneNumRegex,file)
print(fall)
```