Python regex `search` function

Python Regex search() function explained with examples.

We'll cover the following

The Search Function

The `search` function searches for first occurance of a `re` pattern to string with optional flags.

Here is the syntax for this function –

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

Where, `pattern` is the regular expression to be matched, `string` is the string to be searched to match the pattern at the beginning of string and `flags`, which you can specify different flags using bitwise OR (`|`).

Match Flags

Modifier	Description
`re.I`	Performs case-insensitive matching

	matching.
`re.L`	Interprets words according to the current locale. This interpretation affects the alphabetic group (`\w` and `\W`), as well as word boundary behavior (`\b` and `\B`).
`re.M`	Makes `$` match the end of a line and makes ^ match the start of any line.
`re.S`	Makes a period (dot) match any character, including a newline.
`re.U`	Interprets letters according to the Unicode character set. This flag affects the behavior of `\w`, `\W`, `\b`, `\B`.
`re.X`	It ignores whitespace (except inside a set `[]` or when escaped by a `backslash` and treats unescaped `#` as a comment marker.

Return values

- The `re.search` function returns a match `object` on **success** and `None` upon failure. -
- Use `group(n)` or `groups()` function of match object to get matched expression, e.g., `group(n=0)` returns entire match (or specific subgroup `n`)
- The function `groups()` returns all matching subgroups in a tuple (empty if there weren't any).

Example 1

Let's find the words before and after the word `to`:

```
#!/usr/bin/python
import re

line = "Learn to Analyze Data with Scientific Python";

m = re.search( r'(.*) to (.*?) .*', line, re.M|re.I)

if m:
   print "m.group() : ", m.group()
   print "m.group(1) : ", m.group(1)
   print "m.group(2) : ", m.group(2)
else:
   print "No match!!"
```

The first group `(.*)` identified the string: Learn and the next group `(*.?)` identified the string: Analyze.

Example 2

`groups([default])` returns a tuple containing all the subgroups of the match, from 1 up to however many groups are in the pattern.

```
#!/usr/bin/python
import re

line = "Learn Data, Python";

m = re.search( r'(\w+) (\w+)', line, re.M|re.I)

if m:
   print "m.group() : ", m.groups()
   print "m.group (1,2)", m.group(1, 2)
else:
   print "No match!!"
```

Example 3

`groupdict([default])` returns a dictionary containing all the named subgroups of the match, keyed by the subgroup name.

```
#!/usr/bin/python
```

```
import re

number = "124.13";

m = re.search( r'(?P<Expotent>\d+)\.(?P<Fraction>\d+)', number)

if m:
   print "m.groupdict() : ", m.groupdict()
else:
   print "No match!!"
```

Example 4

`start([group])` and `end([group])` return the indices of the start and end of the substring matched by `group`. We need to use `search` instead of `match` for this example:

```
#!/usr/bin/python
import re

email = "hello@leremove_thisarntoanalayzedata.com ";

# m = re.match ("remove_this", email) // This will not work!
m = re.search("remove_this", email)

if m:
   print "email address : ", email[:m.start()] + email[m.end():]
else:
   print "No match!!"
```

Why we need to use `search` instead of the `match`?

Can you guess from the example above?

Show Hint

See in the next lesson!