Compilation Flags

There are 7 compilation flags included in Python 3 that can change how your compiled pattern behaves. Let's go over what each of them do and then we will look at how to use a compilation flag.

re.A / re.ASCII

The ASCII flag tells Python to only match against ASCII instead of using full Unicode matching when coupled with the following escape codes: w, W, b, B, d, D, s and S. There is a re.U / re.UNICODE flag too that is for backwards compatibility purposes; however those flags are redundant since Python 3 already matches in Unicode by default.

re.DEBUG

This will display debug information about your compiled expression.

re.I / re.IGNORECASE

If you'd like to perform case-insensitive matching, then this is the flag for you. If your expression was `[a-z]` and you compiled it with this flag, your pattern will also match uppercase letters too! This also works for Unicode and it's not affect by the current locale.

re.L / re.LOCALE

Make the escape codes: w, W, b, B, d, D, s and S depend on the current locale. However, the documentation says that you should not depend on this flag because the locale mechanism itself is very unreliable. Instead, just use Unicode matching. The documentation goes on to state that this flag really only makes sense for bytes patterns.

re.M / re.MULTILINE

When you use this flag, you are telling Python to make the ^ pattern character match at both the beginning of the string and at the beginning of each line. It also tells Python that *$* should match at the end of the string and the end of each line, which is subtly different from their defaults. See the documentation for additional information.

re.S / re.DOTALL

This fun flag will make the . (period) metacharacter match any character at all. Without the flag, it would match anything except a newline.

re.X / re.VERBOSE

If you find your regular expressions hard to read, then this flag could be just what you need. It will allow to visually separate logical sections of your regular expressions and even add comments! Whitespace within the pattern will be ignored except when in a character class or when the whitespace is preceded by an unescaped backslash.

Using a Compilation Flag

Let's take a moment and look at a simple example that uses the VERBOSE compilation flag! One good example is to take a common email finding regular expression such as r'[w.-]+@[w.-]+' and add some comments using the VERBOSE flag. Let's take a look:

```
import re
re.compile('''
           [\w\.-]+      # the user name
           @
           [\w\.-]+'     # the domain
           ''',
           re.VERBOSE)
```

Let's move on and learn how to find multiple matches.