The main parts of the syntax you will need to know for declaring the pattern matches are:

| **Symbol** | **Meaning** |
| --- | --- |
| ^ | Match the beginning of the text |
| $ | Match the end of the text |
| \d | Match a digit (0, 1, 2, ... 9) |
| \w | Match a word character, e.g. any upper- or lower-case character in the alphabet, digit or the underscore character (\_) |
| + | Match one or more of the preceding character. For example, to match one or more digits you would use \d+. To match one or more "a" characters, you could use a+ |
| \* | Match zero or more of the preceding character. For example, to match nothing or a word you could use \w\* |
| ( ) | Capture the part of the pattern inside the brackets. Any captured values will be passed to the view as unnamed parameters (if multiple patterns are captured, the associated parameters will be supplied in the order that the captures were declared). |
| (?P<*name*>...) | Capture the pattern (indicated by ...) as a named variable (in this case "name"). The captured values are passed to the view with the name specified. Your view must therefore declare an argument with the same name! |
| [  ] | Match against one character in the set. For example, [abc] will match on 'a' or 'b' or 'c'. [-\w] will match on the '-' character or any word character. |

Most other characters can be taken literally!

Let's consider a few real examples of patterns:

| **Pattern** | **Description** |
| --- | --- |
| **r'^book/(?P<pk>\d+)$'** | This is the RE used in our URL mapper. It matches a string that has book/ at the start of the line (**^book/**), then has one or more digits (\d+), and then ends (with no non-digit characters before the end of line marker).  It also captures all the digits **(?P<pk>\d+)** and passes them to the view in a parameter named 'pk'. **The captured values are always passed as a string!**  For example, this would match book/1234 , and send a variable pk='1234' to the view. |
| **r'^book/(\d+)$'** | This matches the same URLs as the preceding case. The captured information would be sent as an unnamed argument to the view. |
| **r'^book/(?P<stub>[-\w]+)$'** | This matches a string that has book/ at the start of the line (**^book/**), then has one or more characters that are *either* a '-' or a word character (**[-\w]+**), and then ends. It also captures this set of characters and passes them to the view in a parameter named 'stub'.  This is a fairly typical pattern for a "stub". Stubs are URL-friendly word-based primary keys for data. You might use a stub if you wanted your book URL to be more informative. For example /catalog/book/the-secret-garden rather than /catalog/book/33. |