6. EXAMPLES 2

Examples with the Question Mark

Introduction

As we have already seen the Question Mark is ***matched if there is zero or one instances of the preceding character (or grouping) in a Regular Expression***.

And we will remember that if we are looking for either the word "colour" or "color", then we can match these using the Question Mark as follows:

```
RegEx_Pattern = "colou?r"
```

So the character preceding the Question Mark ("u") can appear zero or one times.

And we can do the same for multiple characters with the round brackets as follows:

```
RegEx_Pattern = "Reg(ular )?Ex(pression)?"
```

So it will match with either "Regular Expression" or "RegEx".

Some Common Examples

If we are looking for the singular or plural of a word, e.g. "dog" or "dogs", we can do:

```
RegEx_Pattern = "dogs?"
```

So the character preceding the Question Mark ("s") can appear zero or one times.

If we are looking for a word with a prefix or not, e.g. "dependent" or "independent":

```
RegEx_Pattern = "(in)?dependent
```

So the characters preceding the Question Mark ("in") can appear zero or one times.

Greedy and Lazy Matching

The question mark ("?") has additional meanings in Regular Expressions, and one relates to how much of a particular String matches to a regular expression, and this is called *greedy* and *lazy* matching. So, we can define them as follows:

- **Greedy Matching**: The regex will match to as much of the String as possible.
- **Lazy Matching**: The regex will match to as little of the String as possible.

So, for example, if we had a String "`XLazyXGreedyX`", we could match using:

```
RegEx_Pattern = "[a-zA-Z]*"
```

So, this matches any number of characters ("`*`") which are uppercase or lowercase characters ("`a-zA-Z`"). However, this does match a lot of other Strings, so if we wanted to make the search a bit more specific to our string, we could state that the string we are looking for starts with an "`X`" and ends with an "`X`", as follows:

```
RegEx_Pattern = "X[a-zA-Z]*X"
```

And this will match to the entire String "`XLazyXGreedyX`", because a RegEx will typically try to match to as much of a String as possible (it's *greedy* by default), but we can tell the pattern to do a *lazy* match instead, and just match with as much of the String as needed to get a match (so to be *lazy*) using the question mark:

```
RegEx_Pattern = "X[a-zA-Z]*?X"
```

And this will match to the String "`XLazyX`" (the first substring to match the RegEx).