Regular Expressions in Python: A Beginner's Guide

Introduction

Regular expressions (regex) are a powerful tool for pattern matching and manipulation in strings. In Python, the `re` module provides support for regular expressions.

Lesson 1: Importing the `re` Module

Before we can use regular expressions in Python, we need to import the `re` module. This is done with the following line of code:

```python
import re
```

Lesson 2: Basic Regex Symbols

A regular expression is a sequence of characters that forms a search pattern. This pattern can be used with various functions to search, edit, or manipulate text.

Here are some basic regex symbols:

- `.` : Matches any character except newline
- `^` : Matches the start of the string
- `$` : Matches the end of the string
- `*` : Matches 0 or more repetitions
- `+` : Matches 1 or more repetitions
- `{n}` : Matches exactly n repetitions
- `[abc]` : Matches either a, b or c
- `\` : Used to escape special characters

Lesson 3: Using `re.match()`

The `re.match()` function checks if a string starts with a specified pattern. If the pattern matches, `re.match()` returns a match object. If not, it returns `None`.

Here's an example:

```python
import re

string = "Hello, world!"
pattern = "^Hello"

match = re.match(pattern, string)

if match:
    print("Match found!")
else:
    print("No match found.")
```

In this example, the pattern `^Hello` matches any string that starts with "Hello". Since our string "Hello, world!" does start with "Hello", `re.match()` returns a match object and "Match found!" is printed.

Lesson 4: Using `re.search()`

The `re.search()` function searches the entire string for a specified pattern. If the pattern matches, `re.search()` returns a match object. If not, it returns `None`.

Here's an example:

```python
import re

string = "Hello, world!"
pattern = "world"

match = re.search(pattern, string)

if match:
    print("Match found!")
else:
    print("No match found.")
```

In this example, the pattern `world` matches any string that contains "world". Since our string "Hello, world!" does contain "world", `re.search()` returns a match object and "Match found!" is printed.

Lesson 5: Using `re.findall()`

The `re.findall()` function returns all non-overlapping matches of a pattern in a string as a list of strings.

Here's an example:

```python
import re

string = "Hello, world! The world is round."
pattern = "world"

matches = re.findall(pattern, string)

print(matches)  # Outputs: ['world', 'world']
```

In this example, the pattern `world` matches any string that contains "world". Since our string "Hello, world! The world is round." contains "world" twice, `re.findall()` returns a list with two "world" strings.

The good stuff (extraction)

You can extract specific parts of a string using regular expressions in Python by using groups. Groups are created by surrounding the part of the regular expression you want to group with parentheses `()`.

Here's an example:

```python
import re

string = "John Doe, born 1990"
pattern = "(\w+) Doe, born (\d+)"

match = re.search(pattern, string)

if match:
    print("Full Name: ", match.group(0))  # Outputs: John Doe, born 1990
    print("First Name: ", match.group(1))  # Outputs: John
    print("Birth Year: ", match.group(2))  # Outputs: 1990
```

In this example, the pattern `(\w+) Doe, born (\d+)` contains two groups: `(\w+)` and `(\d+)`. The first group matches one or more word characters (equivalent to `[a-zA-Z0-9_]`), and the second group matches one or more digits (equivalent to `[0-9]`).

The `re.search()` function returns a match object if the pattern matches the string. You can then use the `group()` method on the match object to access the groups. `group(0)` returns the entire match, `group(1)` returns the first group, `group(2)` returns the second group, and so on.

Example 1: Extracting Date Components

```python
import re

date_string = "Today's date is 2022-09-30."
pattern = "(\d{4})-(\d{2})-(\d{2})"

match = re.search(pattern, date_string)

if match:
    print("Year: ", match.group(1))  # Outputs: 2022
    print("Month: ", match.group(2))  # Outputs: 09
    print("Day: ", match.group(3))  # Outputs: 30
```

In this example, the pattern `(\d{4})-(\d{2})-(\d{2})` contains three groups that match the year, month, and day in a date string in the format YYYY-MM-DD.

Example 2: Extracting URL Components

```
import re

url_string = "https://www.example.com/path/to/page?query=python"
pattern = "(https?)://(www\.[\w\.]+)/([\w/]+)\?query=(\w+)"

match = re.search(pattern, url_string)

if match:
    print("Protocol: ", match.group(1))  # Outputs: https
    print("Domain: ", match.group(2))  # Outputs: www.example.com
    print("Path: ", match.group(3))  # Outputs: path/to/page
    print("Query: ", match.group(4))  # Outputs: python
```

In this example, the pattern `(https?)://(www\.[\w\.]+)/([\w/]+)\?query=(\w+)` contains four groups that match the protocol, domain, path, and query in a URL.

Remember, these are just basic patterns and might not cover all possible cases. Always tailor your regex to your specific needs.