Regular Expressions in Python: Takeaways

Syntax

REGULAR EXPRESSION MODULE

- Importing the regular expression module:

```
import re
```

- Searching a string for a regex pattern:

```
re.search(r"blue", "Rhythm and blues")
```

- Ignoring capitalization in a regex:

```
string = "My name is Aaron."

re.search(r"aaron", string, flags=re.I)
```

- Replacing a substring:

```
string = "aBcDEfGHIj"

re.sub(r"[A-Z]", "-", string) # returns "a-c--f---j"
```

MATCH OBJECTS

- Extracting the text of a match object without capture groups:

```
match.group()
```

- Extracting the capture groups of a match object with capture groups:

```
match.groups()
```

ESCAPING CHARACTERS

- Treating special characters as ordinary text using backslashes:

```
\[pdf\]
```

Concepts

- Regular expressions, often referred to as regex, are a set of syntax components used for matching sequences of characters in strings.
- A pattern is described as a regular expression that we've written. We say regular expression has matched if it finds the pattern exists in the string.
- `re.search` returns a match object if the pattern is found in the string.
- Character classes allow us to match certain classes of characters.
- A set contains two or more characters that can match in a single character's position.
- Quantifiers specify how many of the previous characters the pattern requires.
- Capture groups allow us to specify one or more groups within our match that we can access separately.
- Negative character classes are character classes that match every character except a character class.
- An anchor matches something that isn't a character, as opposed to character classes which match specific characters.
- A word boundary matches the space between a word character and a non-word character, or a word character and the start/end of a string
- Lookarounds let us define a positive or negative match before or after our string.
- Backreferences allow us to repeat a capture group within our regex pattern.
- Common character classes:

Character Class	**Pattern**	**Explanation**
Set	`[fud]`	Either `f`, `u`, or `d`
Range	`[a-e]`	Any of the characters `a`, `b`, `c`, `d`, or `e`
Range	`[0-3]`	Any of the characters `0`, `1`, `2`, or `3`
Range	`[A-Z]`	Any uppercase letter
Set + Range	`[A-Za-z]`	Any uppercase or lowercase character
Digit	`\d`	Any digit character (equivalent to `[0-9]`)

- Common quantifiers:

Quantifier	Pattern	Explanation
Zero or more	`a*`	The character `a` zero or more times
One or more	`a+`	The character `a` one or more times
Optional	`a?`	The character `a` zero or one times
Numeric	`a{3}`	The character a three times
Numeric	`a{3,5}`	The character `a` three, four, or five times
Numeric	`a{,3}`	The character `a` one, two, or three times
Numeric	`a{8,}`	The character `a` eight or more times

- Common negative character classes:

Character Class	Pattern	Explanation
Negative Set	`[^fud]`	Any character except `f`, `u`, or `d`
Negative Set	`[^1-3Z\s]`	Any characters except `1`, `2`, `3`, `Z`, or whitespace characters
Negative Digit	`\D`	Any character except digit characters
Negative Word	`\W`	Any character except word characters
Negative Whitespace	`\S`	Any character except whitespace characters

- Common anchors:

Anchor	Pattern	Explanation
Beginning	`^abc`	Matches `abc` only at the start of a string
End	`abc$`	Matches `abc` only at the end of a string
Word boundary	`s\b`	Matches `s` only when it's followed by a word boundary
Word boundary	`s\B`	Matches `s` only when it's not followed by a word boundary

- Lookarounds:

Lookaround	Pattern	Explanation
Positive Lookahead	`zzz(?=abc)`	Matches zzzz only when it is followed by `abc`
Negative Lookahead	`zzz(?!abc)`	Matches `zzz` only when it is not followed by `abc`
Positive Lookbehind	`(?<=abc)zzz`	Matches `zzz` only when it is preceded by `abc`
Negative	`(?<!zzz)abc`	Matches `zzz` only when it is not preceded by `abc`

Resources

- re module
- Building regular expressions