Python Regular Expressions (Regex) Cheat Sheet

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.

Basic 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
- `?` : Matches 0 or 1 repetitions
- `{n}` : Matches exactly n repetitions
- `{n,}` : Matches n or more repetitions
- `{,n}` : Matches up to n repetitions
- `{n,m}` : Matches at least n and at most m repetitions
- `[abc]` : Matches either a, b or c
- `[^abc]` : Matches any character except a, b or c
- `a|b` : Matches either a or b
- `()` : Defines a group
- `\` : Used to escape special characters

Special Sequences

- `\d` : Matches any decimal digit; equivalent to `[0-9]`
- `\D` : Matches any non-digit character; equivalent to `[^0-9]`
- `\s` : Matches any whitespace character; equivalent to `[ \t\n\r\f\v]`
- `\S` : Matches any non-whitespace character; equivalent to `[^ \t\n\r\f\v]`
- `\w` : Matches any alphanumeric character; equivalent to `[a-zA-Z0-9_]`
- `\W` : Matches any non-alphanumeric character; equivalent to `[^a-zA-Z0-9_]`
- `\b` : Matches the empty string, but only at the beginning or end of a word
- `\B` : Matches the empty string, but not at the beginning or end of a word

`re` Module Functions

- `re.match(pattern, string)`: Determines if the regex matches at the beginning of the string
- `re.search(pattern, string)`: Searches the string for a match to the regex
- `re.findall(pattern, string)`: Returns all non-overlapping matches of the regex as a list of strings
- `re.sub(pattern, repl, string)`: Replaces all matches of the regex in the string with repl

Using the Built-In `help()` Function

Python's built-in `help()` function is a great way to get help on Python objects, including modules, functions, classes, etc. You can use it in the Python interpreter like this:

```
import re
help(re)
```

This will print out a lot of information about the `re` module, including a brief description of the module, a list of functions and classes provided by the module, and a detailed description of each function and class.

You can also get help on a specific function or class like this:

```
help(re.match)
```

This will print out a detailed description of the `re.match()` function, including its arguments and return value, and a brief example of how to use it.

Examples

Example 1: Validating Email Addresses

```
import re

def is_valid_email(email):
    pattern = r"^[a-zA-Z0-9_.+-]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-.]+$"
    return bool(re.match(pattern, email))

print(is_valid_email("test@example.com"))  # Outputs: True
print(is_valid_email("invalid_email"))  # Outputs: False
```

Example 2: Extracting All URLs from a Text

```
import re

def extract_urls(text):
    pattern = r"http[s]?://(?:[a-zA-Z]|[0-9]|[$-_@.&+]|[!*\\(\\),]|(?:%[0-9a-fA-F][0-9a-fA-F]))+"
    return re.findall(pattern, text)

text = "For more info, visit https://www.example.com or http://example.net"
print(extract_urls(text))  # Outputs: ['https://www.example.com', 'http://example.net']
```

Example 3: Replacing Phone Numbers with a Placeholder

```
import re

def redact_phone_numbers(text):
    pattern = r"\b\d{3}[-.]?\d{3}[-.]?\d{4}\b"
    return re.sub(pattern, "[redacted]", text)

text = "Call me at 555-123-4567 or 555.123.4568"
print(redact_phone_numbers(text))  # Outputs: "Call me at [redacted] or [redacted]"
```

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