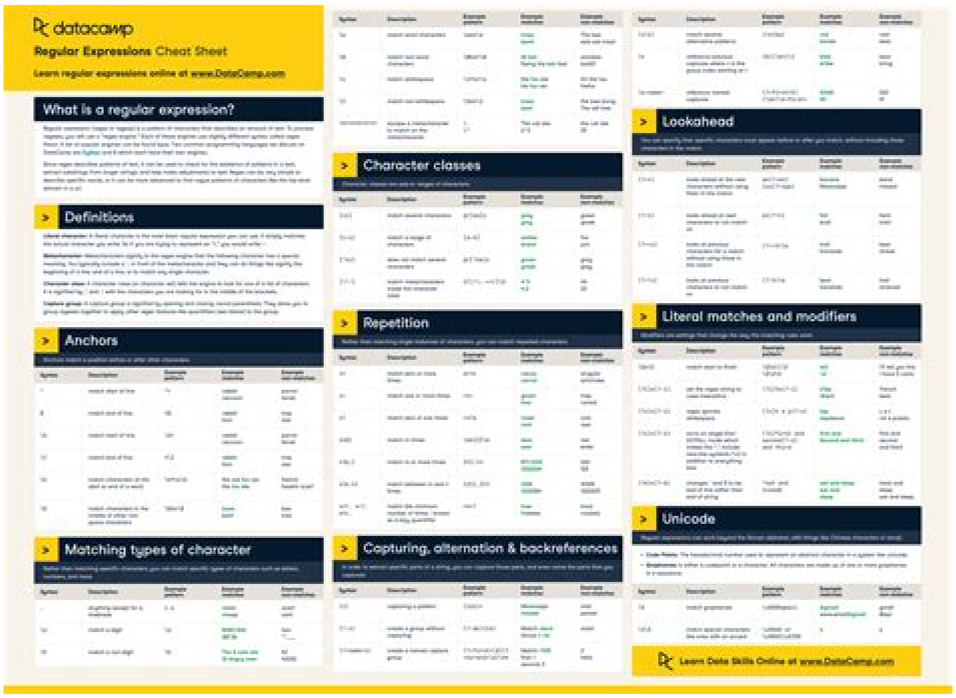
Python Regular Expressions Cheat Sheet


datacamp

Regular Expressions Cheat Sheet

Learn regular expressions online at www.DataCamp.com

What is a regular expression?

Definitions

Anchors

Matching types of character

Character classes

Repetition

Capturing, alternation & backreferences

Lookahead

Literal matches and modifiers

Unicode

Learn Data Skills Online at www.DataCamp.com

Python Regular Expressions Cheat Sheet: Regular expressions (regex) are a powerful tool for string manipulation and pattern matching in Python. They allow developers to search, match, and manage strings with precision and efficiency. This cheat sheet will serve as a comprehensive guide to help you understand and utilize regular expressions in your Python projects effectively. Below, we will cover the essential components of regex, common patterns, methods, and practical examples.

Understanding Regular Expressions

Regular expressions are sequences of characters that form a search pattern. They can be used for various tasks, including:

- Validating input data (e.g., email addresses, phone numbers)
- Searching for specific patterns in text
- Replacing substrings within a string
- Splitting strings based on specific delimiters

Basic Syntax

Here are some crucial elements of Python regex syntax:

- Literal Characters: Match the exact characters. For example, the regex `cat` matches the string "cat".

- Metacharacters: Special characters that have specific meanings:

- `.`: Matches any single character except newline.
- `^`: Matches the start of a string.
- `$`: Matches the end of a string.
- ``: Matches zero or more occurrences of the preceding element.
- `+`: Matches one or more occurrences of the preceding element.
- `?`: Matches zero or one occurrence of the preceding element.
- `{n}`: Matches exactly n occurrences of the preceding element.
- `{n,}`: Matches n or more occurrences of the preceding element.
- `{n,m}`: Matches between n and m occurrences of the preceding element.

- Character Classes: Defines a set of characters to match:
- `[abc]`: Matches any one of the characters a, b, or c.
- `[^abc]`: Matches any character not in the set.
- `[a-z]`: Matches any lowercase letter from a to z.
- `[0-9]`: Matches any digit.

- Groups and Ranges: Use parentheses to create groups for capturing:
- `(abc)`: Captures "abc" as a group.
- `(?:abc)`: Non-capturing group.
- `(?Pabc)`: Named capturing group.

Common Patterns

Regular expressions can be used to match specific types of data. Here are some common patterns:

Email Address

A basic regex for validating an email address might look like this:

```python
^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$
```

- Explanation:
- `^`: Start of the string.
- `[a-zA-Z0-9._%+-]+`: One or more characters that are alphanumeric or special characters.
- `@`: The "@" symbol.
- `[a-zA-Z0-9.-]+`: Domain name consisting of alphanumeric characters, dots, or hyphens.
- `\.`: A literal dot.
- `[a-zA-Z]{2,}`: Domain suffix of at least two letters.
- `$`: End of the string.

Phone Number

A regex pattern for validating a phone number could be:

```python
^\+?1?\d{10}$
```

- Explanation:
- `^`: Start of the string.
- `\+?`: Optional "+" for international format.
- `1?`: Optional country code for the USA.
- `\d{10}`: Exactly 10 digits.
- `$`: End of the string.

URL

To validate a URL, you can use:

```python
^(http|https)://[a-zA-Z0-9.-]+(\.[a-zA-Z]{2,})?(/.)?$
```

- Explanation:
- `^`: Start of the string.
- `(http|https)://`: Matches either "http://" or "https://".
- `[a-zA-Z0-9.-]+`: Domain name.
- `(\.[a-zA-Z]{2,})?`: Optional top-level domain.
- `(/.)?`: Optional path.
- `$`: End of the string.

Using Regular Expressions in Python

To work with regular expressions in Python, you need to import the `re` module. The module provides several essential functions:

Basic Functions

1. re.match(): Determines if the regular expression matches at the beginning of a string.
```python
import re
result = re.match(r'abc', 'abcdef')
```

2. re.search(): Searches the string for any location where the regex matches.
```python
result = re.search(r'abc', '123abcdef')
```

3. re.findall(): Returns a list of all non-overlapping matches in the string.
```python
result = re.findall(r'\d+', 'There are 12 apples and 24 oranges')
```

4. re.sub(): Replaces occurrences of the regex pattern with a specified string.
```python
result = re.sub(r'apples', 'bananas', 'I have apples and oranges')
```

5. re.split(): Splits the string at each match of the regex.
```python
result = re.split(r'\s+', 'Hello World')
```

Flags

Flags modify the behavior of regex matching. Common flags include:

- re.IGNORECASE: Makes the matching case-insensitive.
- re.MULTILINE: Allows `^` and `$` to match the start and end of each line.
- re.DOTALL: Makes `.` match any character, including newline.

Example usage of flags:

```python
result = re.findall(r'\d+', 'A1 B2 C3', re.IGNORECASE)
```

Practical Examples

Let’s explore some practical examples to demonstrate the use of regex in Python.

Example 1: Extracting Dates

To extract dates in the format `DD/MM/YYYY`, you can use:

```python
import re

text = "Today's date is 25/12/2023."
pattern = r'(\d{2})/(\d{2})/(\d{4})'
matches = re.findall(pattern, text)

for match in matches:
```

```
print("Day:", match[0], "Month:", match[1], "Year:", match[2])
```

Example 2: Validating Passwords

A password must meet the following criteria: at least 8 characters long, contains both uppercase and lowercase letters, at least one numeric digit, and at least one special character.

```python
import re

password = "Password123!"
pattern = r'^(?=.[a-z])(?=.[A-Z])(?=.\d)(?=.[@$!%?&])[A-Za-z\d@$!%?&]{8,}$'
is_valid = re.match(pattern, password)

if is_valid:
print("Password is valid.")
else:
print("Password is invalid.")
```

Example 3: Cleaning Text

You can use regex to clean unwanted characters from a string:

```python
import re

text = "Hello, World!!! @2023"
cleaned_text = re.sub(r'[^\w\s]', '', text)
print(cleaned_text) Output: Hello World 2023
```

Conclusion

The Python Regular Expressions Cheat Sheet outlined the fundamental components, common patterns, and practical applications of regex in Python. Mastering regular expressions will significantly enhance your ability to manipulate and validate strings in your projects. With practice, you will find regex to be an invaluable tool in your programming toolkit. Whether you are validating user inputs, extracting information from text, or performing complex string manipulations, regular expressions can streamline the process and make your code more efficient.

Frequently Asked Questions

What are Python regular expressions used for?

Python regular expressions are used for searching, matching, and manipulating strings based on specific patterns.

How do you import the regular expressions module in Python?

You can import the regular expressions module in Python by using the statement 'import re'.

What does the '.' character represent in a regular expression?

In a regular expression, the '.' character matches any single character except a newline.

How do you match a digit using Python regular expressions?

You can match a digit using the '\d' pattern, which represents any digit from 0 to 9.

What is the purpose of the '^' and '$' anchors in regular expressions?

The '^' anchor asserts the position at the start of a string, while the '$' anchor asserts the position at the end of a string.

How can you use regex to find all occurrences of a pattern in a string?

You can use the 're.findall()' function to find all occurrences of a pattern in a string, returning them as a list.

[Python Regular Expressions Cheat Sheet]()

What does colon equal (:=) in Python mean? - Stack Overflow
Mar 21, 2023 · In Python this is simply =. To translate this pseudocode into Python you would need to know the data structures being referenced, and a bit more of the algorithm …

*What does asterisk * mean in Python? - Stack Overflow*
What does asterisk * mean in Python? [duplicate] Asked 16 years, 7 months ago Modified 1 year, 6 months ago Viewed 319k times

What does the "at" (@) symbol do in Python? - Stack Overflow
Jun 17, 2011 · 96 What does the “at” (@) symbol do in Python? @ symbol is a syntactic sugar python provides to utilize decorator, to paraphrase the question, It's exactly about what does ...

Is there a "not equal" operator in Python? - Stack Overflow
Jun 16, 2012 · 1 You can use the != operator to check for inequality. Moreover in Python 2 there was <> operator which used to do the same thing, but it has been deprecated in Python 3.

Using or in if statement (Python) - Stack Overflow
Using or in if statement (Python) [duplicate] Asked 7 years, 6 months ago Modified 8 months ago Viewed 149k times

python - What is the purpose of the -m switch? - Stack Overflow
Python 2.4 adds the command line switch -m to allow modules to be located using the Python module namespace for execution as scripts. The motivating examples were standard library ...

What is Python's equivalent of && (logical-and) in an if-statement?
Mar 21, 2010 · There is no bitwise negation in Python (just the bitwise inverse operator ~ - but that is not equivalent to not). See also 6.6. Unary arithmetic and bitwise/binary operations and 6.7. ...

syntax - What do >> and <
Apr 3, 2014 · 15 The other case involving print >>obj, "Hello World" is the "print chevron" syntax for the print statement in Python 2 (removed in Python 3, replaced by the file argument of the ...

python - Is there a difference between "==" and "is"? - Stack ...
Since is for comparing objects and since in Python 3+ every variable such as string interpret as an object, let's see what happened in above paragraphs. In python there is id function that shows ...

*python - What does ** (double star/asterisk) and * (star/asterisk) ...*
Aug 31, 2008 · A Python dict, semantically used for keyword argument passing, is arbitrarily ordered. However, in Python 3.6+, keyword arguments are guaranteed to remember insertion ...

What does colon equal (:=) in Python mean? - Stack Overflow
Mar 21, 2023 · In Python this is simply =. To translate this pseudocode into Python you would need to know the data structures being referenced, and a bit more of the algorithm implementation. Some notes about psuedocode: := is the assignment operator or = in Python = is the equality operator or == in Python There are certain styles, and your mileage may vary:

What does asterisk * mean in Python? - Stack Overflow
*What does asterisk * mean in Python? [duplicate] Asked 16 years, 7 months ago Modified 1 year, 6 months ago Viewed 319k times*

What does the "at" (@) symbol do in Python? - Stack Overflow
Jun 17, 2011 · 96 What does the “at” (@) symbol do in Python? @ symbol is a syntactic sugar python provides to utilize decorator, to paraphrase the question, It's exactly about what does decorator do in Python? Put it simple decorator allow you to modify a given function's definition without touch its innermost (it's closure).

Is there a "not equal" operator in Python? - Stack Overflow
Jun 16, 2012 · 1 You can use the != operator to check for inequality. Moreover in Python 2 there was <> operator which used to do the same thing, but it has been deprecated in Python 3.

Using or in if statement (Python) - Stack Overflow
Using or in if statement (Python) [duplicate] Asked 7 years, 6 months ago Modified 8 months ago Viewed 149k times

python - What is the purpose of the -m switch? - Stack Overflow
Python 2.4 adds the command line switch -m to allow modules to be located using the Python module namespace for execution as scripts. The motivating examples were standard library modules such as pdb and profile, and the Python 2.4 implementation is ...

What is Python's equivalent of && (logical-and) in an if-statement?
Mar 21, 2010 · There is no bitwise negation in Python (just the bitwise inverse operator ~ - but that is not equivalent to not). See also 6.6. Unary arithmetic and bitwise/binary operations and 6.7. Binary arithmetic operations. The logical operators (like in many other languages) have the advantage that these are short-circuited.

syntax - What do >> and <
Apr 3, 2014 · 15 The other case involving print >>obj, "Hello World" is the "print chevron" syntax for the print statement in Python 2 (removed in Python 3, replaced by the file argument of the print() function). Instead of writing to standard output, the output is passed to the obj.write() method. A typical example would be file objects having a write() method.

python - Is there a difference between "==" and "is"? - Stack ...
Since is for comparing objects and since in Python 3+ every variable such as string interpret as an object, let's see what happened in above paragraphs. In python there is id function that shows a unique constant of an object during its lifetime. This id is using in back-end of Python interpreter to compare two objects using is keyword.

python - What does ** (double star/asterisk) and * (star/asterisk) ...
Aug 31, 2008 · A Python dict, semantically used for keyword argument passing, is arbitrarily ordered. However, in Python 3.6+, keyword arguments are guaranteed to remember insertion order.

Unlock the power of Python with our comprehensive regular expressions cheat sheet! Discover how to master regex patterns efficiently. Learn more now!

Back to Home