Python Data Collection and Management for Public Policy Research

Day 3: `git` and Github, Regular Expressions, Getting Set Up

Blake Miller[†]

3 July, 2024

[†]Assistant Professor, Department of Methodology, London School of Economics and Political Science (E-mail: b.a.miller@lse.ac.uk)

Agenda for Today

- Quick Overview of Github reference materials
- Coding Session: The git/GitHub workflow
- Reviewing Quiz 1
- Regular Expressions
- Coding Session: Searching in Sublime Text with Regular Expressions
- Installing Python
- Connecting Sublime Text and the CLI
- First look at the Python shell

Quick Overview of Github Guides

Github Guides

- Please use these Github Guides as a reference.
- You can access them on the course page here here.

Coding Session: The git/GitHub workflow

Learning the Workflow through Poem Translation

- Since we don't yet know much code, let's use the languages we know.
- We can start with a famous poem by Maya Angelou: Still I Rise, by Maya Angelou.

Learning the `git`/Github Workflow through Poem Translation

Instructions:

1. Tell me your GitHub account name and I will add you as a collaborator to the project and assign you a stanza number.
2. Go to `day_3/02_github_exercise.md` in the course Github page.

Regular Expressions

What are Regular Expressions?

- Regular expressions (regex) are patterns used to match character combinations in text.
- They allow us some flexibility in search so we can find certain kinds of data we might want to extract or quantify.
- Commonly used in data cleaning, data extraction, and complex data analysis.
- Great interactive tutorial on RegexOne

Why Learn Regular Expressions?

- Quickly find and replace patterns in text and data.
- Perform complex text matching and extraction, even when data are non-standard or dirty.
- When collecting data from online, it helps in the parsing of raw data.

Practical Applications of Regex

- Finding specific patterns within data sets, like dates, emails, and phone numbers.
- Cleaning and preparing data for analysis.
- Automating the extraction of structured data from text, such as extracting all hyperlinks from a webpage.

Key Terms in Regular Expressions

Metacharacters Special characters that control the logic of a pattern in regular expressions.

Pattern The format or sequence that a regular expression defines, which is used to match against strings.

Character Class A set of characters enclosed within square brackets `[]` that matches any single character within the brackets. For example, `[abc]` matches "a", "b", or "c".

Key Terms in Regular Expressions

Grouping Parentheses `()` are used to group parts of expressions so that quantifiers or other operations can be applied to the entire group.

Greedy Match The default behavior of quantifiers that capture as much of the string as possible.

Lazy Match Quantifiers followed by a `?` that modify them to capture as little of the string as possible, such as `*?` or `+?`.

Anchors Special metacharacters `^` and `$` that do not match characters but rather the positions before or after characters. Used to match a position before, after, or between characters.

Escape Characters The backslash `\` is used to escape metacharacters so that they are treated as ordinary characters.

Basic Metacharacters

Character	Description
.	Matches any single character except newline.
*	Matches zero or more of the preceding element.
+	Matches one or more of the preceding element.

Special Character Classes

Character	Description
`\d`	Matches any digit (equivalent to `[0-9]`).
`\w`	Matches any word character (alphanumeric & underscore).
`\s`	Matches any whitespace character (spaces, tabs, line breaks).

Using Brackets and Hyphens

Character	Description
`[abc]`	Matches any of ‘a’, ‘b’, or ‘c’.
`[a-z]`	Matches any lowercase letter from ‘a’ to ‘z’.
`[A-Za-z]`	Matches any letter regardless of case.

Grouping in Regular Expressions

Character	Description
( )	Groups parts of the expression. Useful for: • Applying quantifiers to sequences as a single unit. • Capturing substrings for back-referencing. • Using alternation within the group (e.g., `(dog|cat)`).
|	Represents alternation (logical OR), used within groups to match one of several patterns.

Lookahead and Lookbehind in Regular Expressions

Feature	Description
`(?=...)`	**Positive Lookahead:** Asserts that what immediately follows the current position in the string is the pattern specified inside the parentheses, without including it in the match.
`(?!...)`	**Negative Lookahead:** Asserts that what immediately follows the current position in the string is not the pattern specified inside the parentheses.
`(?<=...)`	**Positive Lookbehind:** Asserts that what immediately precedes the current position in the string is the pattern specified inside the parentheses, without including it in the match.
`(?<!...)`	**Negative Lookbehind:** Asserts that what immediately precedes the current position in the string is not the pattern specified inside the parentheses.

Regular Expressions for Non-Latin Characters

You can use regular expressions with non-Latin characters using Unicode ranges. Unicode is the standard text input system used by most computers as a default.

Expression	Description
`[\u4e00-\u9fff]`	Matches any character in the range of common Chinese characters (Unicode range for CJK Unified Ideographs).
`[\u3400-\u4DBF]`	Matches characters in the Unicode range for CJK Unified Ideographs Extension A, less commonly used but still valid Chinese characters.
`[\u20000-\u2A6DF]`	Matches characters in the range of CJK Unified Ideographs Extension B, which includes historical and rare characters.
`[\u2A700-\u2B73F]`	Matches characters in the range of CJK Unified Ideographs Extension C.

Building on Regular Expressions and Git

- Regular Expressions (Regex) Tutorial by Corey Schafer on YouTube
- Interactive Regex Tutorial at RegexOne
- Applications of Regex to Text Processing by Monica Pérez Nogueras
- Git Tutorial for Beginners: Command-Line Fundamentals by Corey Schafer on YouTube

Coding Session: Searching in Sublime Text with Regular Expressions

Installing Python

Connecting Sublime Text and the CLI

Sublime Customization Instructions

- Please use the instructions on the course website.
- Choose the instructions appropriate for your system here.

First look at the Python shell

The Python Shell

Introduction to the Python Shell

- The Python shell, also known as the Python interpreter, is an interactive interface for Python programming.
- It allows for quick testing of Python code snippets and direct interaction with the Python execution environment.

Accessing the Python Shell

- Open a command line interface (CLI) like Terminal on macOS or Command Prompt on Windows.
- Type `python` or `python3` (depending on your installation) and press Enter.

The Python Shell (continued)

Features of the Python Shell

- Immediate feedback for expressions and statements.
- Useful for learning syntax and features of Python.
- Useful tool for developing and debugging Python scripts.
- Supports dynamic type checking, auto-completion, and history of commands.

Example of Using the Python Shell

```
>>> print("Hello, World!")
Hello, World!
```

Let's Try the Python Shell