Woche 7: Fortgeschrittene Programmierung und Programmiertechniken

Einleitung

In dieser Woche liegt der Fokus auf der Erkundung von regulären Ausdrücken, Testen von Anwendungen und Dokumentation - Elemente, die für jeden Programmierer unverzichtbar sind. Dabei sind die Lernmaterialien und Aufgaben darauf ausgerichtet, praktische Fähigkeiten in den Bereichen Textanalyse, Code-Test und Dokumentation zu vermitteln.

Zu Beginn der Woche werden reguläre Ausdrücke und das "re" Modul von Python im Mittelpunkt stehen, um effiziente Mustererkennung und Textbearbeitung zu ermöglichen. Dies wird von großem Nutzen sein, um komplexe Textdaten zu analysieren und zu manipulieren. In der Tat, ein guter Griff auf reguläre Ausdrücke ist ein mächtiges Werkzeug in der Toolbox jedes Entwicklers.

Darüber hinaus wird die Bedeutung des Testens von Anwendungen beleuchtet und erlernt, wie man effektive Unit-Tests mit dem "unittest" Module erstellt. Diese Fähigkeiten sind entscheidend, um die Qualität der Softwareentwicklung zu gewährleisten und sicherzustellen, dass alle Teile des Codes wie erwartet funktionieren.

Schließlich wird das Thema Dokumentation behandelt - ein oft übersehener, aber dennoch entscheidender Aspekt der Softwareentwicklung. Eine gut dokumentierte Anwendung erleichtert anderen Entwicklern (und auch Ihnen selbst in der Zukunft) das Verständnis und die Wartung des Codes.

Gesamtüberblick

Hier ein Überblick über die Inhalte und Aktivitäten der aktuellen Woche:

- Selbststudium:
    - Reguläre Ausdrücke
    - re Modul
    - Testen von Anwendungen
    - Unittest
    - Pytest
    - Dokumentation von Anwendungen
- Aufgabe:
    - Text mit Regex analysieren
    - Unittests anlegen
    - Testbaren Code entwerfen
    - Eine Anwendung dokumentieren

Inhalte und thematische Abgrenzung

Die folgende Auflistung zeigt detailliert, welche Themen Sie in der Woche behandeln und bearbeiten. Sie sind eine Voraussetzung für die folgenden Wochen und sollten gut verstanden worden sein. Wenn es Verständnisprobleme gibt, machen Sie sich Notizen und fragen Sie am Präsenztag nach, so dass wir gemeinsam zu Lösungen kommen können. Und denken Sie bitte immer daran: es gibt keine „dummen“ Fragen!

- Reguläre Ausdrücke (Regex):
    - Grundlagen der Regulären Ausdrücke
    - Metazeichen und Sequenzen
    - Quantifizierer und Gruppierung
    - Verwendung von Flags
    - Anwendungsfälle und Beispiele
- re Modul:
    - Einführung in das re Modul
    - Match und Search Funktionen
    - Findall und Split Funktionen
    - Sub und Subn Funktionen
    - Kompilierung von Regulären Ausdrücken
- Testen von Anwendungen:
    - Einführung in Softwaretests
    - Warum Tests wichtig sind
    - Unterschied zwischen manuellen und automatisierten Tests
    - Arten von Softwaretests (Unit Tests, Integration Tests, Funktionale Tests)
    - Testabdeckung und Testgetriebene Entwicklung (TDD)
- Unittest
    - Einführung in das unittest Modul
    - Testfall und TestSuite
    - Test Runner und Test Loader
    - Schreiben und Ausführen von Unit Tests
    - Verwendung von Assertions
- Dokumentation von Anwendungen:
    - Warum Dokumentation wichtig ist
    - Arten der Dokumentation (Code Kommentare, README, API Dokumentation)
    - Einführung in Python Docstrings
    - Verwendung von Sphinx zur Erstellung von Dokumentationen
    - Automatisierte Dokumentation mit Read the Docs

Lernpfad

Der Lernpfad ist ein Vorschlag, in welcher Reihenfolge Sie die Inhalte der Woche angehen können. Betrachten Sie ihn gerne als eine Todo-Liste, die Sie von oben nach unten abhaken. So können Sie sicher sein, dass Sie alle wichtigen Themen bearbeitet haben und sind gut vorbereitet für die folgenden Wochen.

1. Reguläre Ausdrücke und das re Modul:
   - Beginnen Sie mit dem Studium der Grundlagen von regulären Ausdrücken und wie sie zur Mustererkennung in Textdaten verwendet werden können.
   - Machen Sie sich dann mit dem re Modul in Python vertraut, das für die Arbeit mit regulären Ausdrücken verwendet wird.
   - Lernen Sie die verschiedenen Funktionen und Methoden des Moduls kennen, wie z.B. match, search, findall, split und sub.
   - Üben Sie das Erstellen von Mustern mit speziellen Sequenzen, Zeichenklassen, Quantifikatoren und Metazeichen und wenden Sie diese in Übungen an.
2. Testen von Anwendungen:
   - Beschäftigen Sie sich mit der Wichtigkeit des Testens von Anwendungen.
   - Verstehen Sie, warum Tests für die Qualitätssicherung in der Softwareentwicklung unerlässlich sind.
   - Machen Sie sich mit den verschiedenen Testebenen vertraut, einschließlich UnitTests, Integrationstests, Funktionstests, Systemtests und Akzeptanztests.
   - Lernen Sie, wie Sie Tests für Ihre Anwendungen entwerfen und durchführen können.
3. Unittest:
   - Führen Sie sich in das unittest Framework ein, das in Python für das Schreiben von Unit-Tests verwendet wird.
   - Lernen Sie die Struktur des Frameworks kennen und wie Sie Testfälle und Test-Suites erstellen können.
4. Dokumentation von Anwendungen:
   - Erkunden Sie die Bedeutung einer guten Dokumentation für Ihre Anwendungen.
   - Lernen Sie, wie Sie Dokumentationen schreiben, die sowohl für Sie als auch für andere Entwickler nützlich sind.
   - Erfahren Sie mehr über die verschiedenen Formen der Dokumentation, einschließlich Code-Kommentare, Readme-Dateien und formelle Dokumentation.
5. Übungsaufgaben
   - Schreiben Sie Skripte, die Reguläre Ausdrücke verwenden, um Text zu analysieren und Muster zu erkennen.
   - Erstellen Sie Testfälle für Ihre bisherigen Anwendungen oder Übungen unter Verwendung von Unittest oder pytest.
   - Entwerfen Sie einen Code, der testbar ist, indem Sie ihn in Funktionen und Module aufteilen.
   - Dokumentieren Sie eine Ihrer Anwendungen: Kommentieren Sie den Code, erstellen Sie eine Readme-Datei und eine formelle Dokumentation.
6. Selbstbewertung
7. Gemeinsames Programmieren

Programmieraufgaben

Die folgenden Programmieraufgaben sollen Ihnen eine Anregung geben. Haben Sie eigene Ideen und Themen, die Sie ausprobieren wollen, dann sollten Sie diesen nachgehen. Wichtig ist vor allem, dass Sie „Dinge ausprobieren“. Und auch, dass Sie Fehler machen, sowohl syntaktische als auch semantische. Versuchen Sie diese Fehler zu finden und aufzulösen, dann gerade aus den Fehlern lernen Sie am Ende am meisten.

1. Reguläre Ausdrücke:

Schreiben Sie ein Python-Skript, das mithilfe von regulären Ausdrücken alle Email-Adressen aus einem gegebenen Text extrahiert.

2. Testen von Anwendungen mit Unittest:

Entwickeln Sie eine einfache Funktion, zum Beispiel eine Funktion, die die Fakultät einer Zahl berechnet. Schreiben Sie dann Testfälle für diese Funktion mithilfe des Unittest-Frameworks.

3. Dokumentation:

Wählen Sie eine der in den vorherigen Wochen erstellten Anwendungen aus und dokumentieren Sie sie gründlich. Schreiben Sie Code-Kommentare, erstellen Sie eine Readme-Datei und erstellen Sie eine formelle Dokumentation, die die Verwendung der Anwendung erklärt.

4. Integrationstests:

Erweitern Sie die Anwendung aus Aufgabe 4 um eine weitere Funktion. Schreiben Sie anschließend einen Integrationstest, der sicherstellt, dass beide Funktionen korrekt zusammenarbeiten.

Ressourcen

Hier nun die Verweise auf Lernquellen, die uns für diese Woche und ihre Inhalte geeignet erscheinen. Je nachdem, welcher Lerntyp Sie sind, wählen Sie sich ihre bevorzugte Quelle, es ist nicht zwingend notwendig alle durchgearbeitet zu haben. Allerdings sollten die Inhalte des Lernpfads angesprochen und verstanden worden sein.

- **Buch**: Programming Techniques using Python

  Regular Expressions: Dieses Thema wird in "Kapitel 6: Python Regular Expressions" behandelt.

- **Buch**: Python Crash Course:

  Testen: „Kapitel 11: Testing your Code“

- **Buch**: Python Unit Test Automation

  Alles über Testen

- **Buch**: Publishing Python Package

  Dokumentation: Kapitel 8

  Testen: Kapitel 5

- **Video**:
  - Regular Expressions
  - Regular Expressions in Python
- **Lab and Course**:
  - Course: Get started with testing in Python
  - Course: Python Unit Testing: Testing Python Code Using pytest