bernd KLEIN

Numerisches PYTHON

Arbeiten mit **NumPy**, **Matplotlib** und **Pandas**

HANSER

Programmiersprachen sind wie Schuhe. Es gibt nicht den Schuh für alle Fälle. Einen Schuh, den man sowohl für feierliche Anlässe oder im Büro als auch beim Sport oder bei Wanderungen tragen kann. Python ist jedoch eine Sprache, die sich universell in den meisten Gebieten einsetzen lässt.

Man kann sich nun die berechtigte Frage stellen, worauf dieser große Erfolg von Python beruht. Zu den Hauptgründen für die Beliebtheit und die Verbreitung von Python zählt auf jeden Fall die Benutzung von Python bei „Big Data" und dem „Maschinellen Lernen". Zwei Gebiete, in denen es um die Lösungen von numerischen Problemen geht. Betrachtet man jedoch das reine Python, so wie es Anfang der 90er Jahre designt und implementiert worden ist, stellt man rasch fest, dass diese Sprache zur Lösung von numerischen Problemen eigentlich denkbar ungeeignet ist. Anfänglich lag der Fokus von Python keinesfalls auf der numerischen Programmierung. Was natürlich nicht bedeutet, dass man keine numerischen Probleme lösen konnte. Python stellte mit Listen und Dictionaries fantastische Datenstrukturen zur effizienten Lösung von vielfältigen allgemeinen Problemen dar. Allerdings sind diese Datenstrukturen völlig ungeeignet, numerische Berechnung, wie man sie beispielsweise im maschinellen Lernen benötigt, effizient zu implementieren. Listen sind für diese Aufgaben im Vergleich zu effizienten Array-Implementierungen zum einen extrem langsam und verschwenden zum anderen extreme Mengen von Speicherplatz. So können Array-Implementierungen beispielsweise fünfzig- bis hundertmal so langsam sein und gleichzeitig einen etwa dreißigfach größeren Speicherplatzbedarf haben. Die Lösung hierfür bieten die Module NumPy, SciPy, Matplotlib und Pandas. So stellt NumPy Datenstrukturen zur Verfügung, die um den Faktor 10 bis 100 schneller sind als Implementierungen in reinem Python oder anderen numerisch ungeeigneten Programmiersprachen.

1.2 Aufbau des Buches

In diesem Buch geht es um Python und seine hervorragenden Möglichkeiten zum Einsatz bei numerischen Problemen. Also damit um die Module, die für „Big Data" und das „Maschinelle Lernen" unabdinglich sind. Auch wenn das Buch mit einer kompletten, aber sehr knapp gehaltenen Einführung in die Grundlagen von Python beginnt, sind Grundkenntnisse in Python beim Lesen dieses Buches von großem Vorteil. Grundkenntnisse, wie man sie beispielsweise in meinem Buch „Einführung in Python 3: Für Ein- und Umsteiger"[6] erwerben kann. Die Schwerpunkte des vorliegenden Buches liegen auf den Modulen NumPy, Matplotlib und Pandas. Genau die Module, die einen

wesentlichen Anteil zum steilen Aufstieg von Python im Ranking der beliebtesten Programmiersprachen beigetragen haben.

1.3 Python-Installation

Wir gehen davon aus, dass Python bei den Leserinnen und Lesern des Buches bereits installiert ist, insbesondere mit den genannten Modulen NumPy, Matplotlib und Pandas. Sollte dies nicht der Fall sein, empfehlen wir die Installation von Anaconda[7], die sich auf allen Betriebssystemen äußerst einfach gestaltet. Damit ist alles installiert, was im Laufe des Buches benötigt wird.

1.4 Download der Beispiele

Alle im Buch verwendeten Beispiele finden Sie zum Download unter

http://www.python-kurs.eu/buecher/numerical_python/

Dort findet sich auch ein Korrekturverzeichnis.

1.5 Anregungen und Kritik

Falls Sie glauben, eine Ungenauigkeit oder einen Fehler im Buch gefunden zu haben, können Sie auch gerne eine E-Mail direkt an den Autor schicken: klein@python-kurs.eu.

Natürlich gilt dies auch, wenn Sie Anregungen oder Wünsche zum Buch geben wollen. Leider können wir jedoch – so gerne wir es auch tun würden – keine individuellen Hilfen zu speziellen Problemen geben.

Wir werden versuchen, Fehler und Anmerkungen in kommenden Auflagen zu berücksichtigen. Selbstverständlich aktualisieren wir damit auch unsere Informationen unter

http://www.python-kurs.eu/buecher/numerical_python/

Ich wünsche Ihnen viel Spaß und Erfolg beim Durcharbeiten dieses Buches!

Bernd Klein, Februar 2019

---

[1] https://stackoverflow.blog/2017/10/31/disliked-programming-languages/

[2] Bei dem TIOBE-Index des niederländischen Unternehmens TIOBE Software BV handelt es sich um ein seit 2001 publiziertes und monatlich aktualisiertes Ranking von Programmiersprachen nach ihrer Popularität. Der Index wird jeden Monat aktualisiert. Der Listenplatz einer Sprache ergibt sich aus der Häufigkeit von Treffern bei der Suche nach dem Namen dieser Programmiersprache in den wichtigsten Suchmaschinen wie Google, Bing, Yahoo! und so weiter. Dies bedeutet also nicht, dass es sich um ein Ranking der „besten" Programmiersprachen oder der Sprachen mit den meisten Codezeilen handelt!

[3] https://www.tiobe.com/tiobe-index/

[4] So ist die Laufzeitumgebung des Android-Betriebssystems (ART, Android Runtime) in Java geschrieben.

[5] http://pypl.github.io/PYPL.html

[6] Bernd Klein, Einführung in Python 3: Für Ein- und Umsteiger, Carl Hanser Verlag GmbH & Co. KG; Auflage: 3., überarbeitete (6. November 2017)

[7] https://www.anaconda.com/distribution/

2 Numerisches Programmieren mit Python

2.1 Definition von numerischer Programmierung

Der Titel unseres Buches lautet „Numerisches Python“, was eine Anspielung auf „Numerisches Programmieren“ ist.

Der Ausdruck „numerisches Programmieren“ – auch bekannt unter dem Begriff „wissenschaftliches Programmieren“ – ist irreführend. Man könnte es als eine Programmierung ansehen, die mit Zahlen statt mit z.B. Texten zu tun hat. Letztendlich haben die meisten Programme, auch wenn sie scheinbar nichts mit Zahlen zu tun haben, einen numerischen Kern. Denkt man beispielsweise an den Google-Algorithmus und an die Art, wie er einem auf eine Suchanfrage Vorschläge zu Webseiten offeriert, dann könnte man glauben, dass es sich bei dem zugrunde liegenden Algorithmus um reine Textverarbeitung handelt. Dennoch ist auch in diesem Fall der Kern bzw. der wesentliche Teil des Algorithmus ein numerisches Problem. Umseinen PageRanking-Algorithmus (d.h. die Bewertung der Webseiten) durchzuführen, lässt Google die größte jemals von Menschen erdachte Matrix berechnen.

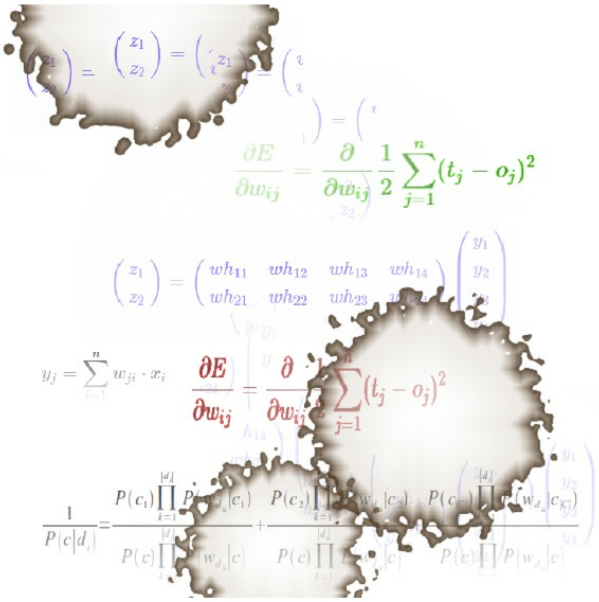
So könnte man denken, dass es sich letztendlich bei jedem Programm um numerische Programmierung handelt, aber es gibt auch eine engere Definition.

Unter numerischer Programmierung versteht man das Gebiet der Informatik und der Mathematik, in dem es um Approximationsalgorithmen geht, d.h. die numerische Approximation von mathematischen Problemen oder numerischer Analysis. In anderen Worten: Probleme mit stetigen Variablen.

In unserem Buch haben wir insbesondere die numerischen Verfahren im Fokus, die in den Gebieten „Data Science“ und „Maschinelles Lernen“ besonders benötigt werden.

Python gehört zu den wichtigsten und am häufigsten benutzten Programmiersprachen in diesem Gebiet. Allerdings würde Python keine Rolle spielen, wenn es nicht mächtige Module zur numerischen Programmierung zur Verfügung stellte, die wir im Folgenden beschreiben werden.

2.2 Zusammenhang zwischen Python, NumPy, Matplotlib, SciPy und Pandas



Python ist eine universelle Programmiersprache, die sich in den unterschiedlichsten Gebieten einsetzen lässt. So zum Beispiel in der Systemadministration, als Tool zur Erzeugung und zum Betrieb von dynamischen Webseiten und in der Computerlinguistik. Da Python ein universelle Programmiersprache ist, lässt sie sich natürlich auch zum Lösen numerischer Probleme einsetzen. So weit so gut, aber die Crux bei der Sache liegt in der Laufzeit und auch im Speicherverbrauch. Reines Python – also ohne den Einsatz irgendwelcher numerischer Spezialmodule – würde sich nicht eignen für Aufgaben, für die Matlab und R geschaffen worden sind. Sobald es um die Lösung numerischer Probleme geht, ist die Leistungsfähigkeit von Algorithmen von höchster Wichtigkeit, sowohl was die Geschwindigkeit als auch den Speicherverbrauch betrifft.