合肥大學
HEFEI UNIVERSITY

Programming with Python

2. Einleitung

Thomas Weise (汤卫思)
tweise@hfuu.edu.cn

Institute of Applied Optimization (IAO)
School of Artificial Intelligence and Big Data
Hefei University
Hefei, Anhui, China

应用优化研究所
人工智能与大数据学院
合肥大学
中国安徽省合肥市

Version: 2025-10-29

Programming with Python

Dies ist ein Kurs über das Programmieren mit der Programmiersprache Python an der Universität Hefei (合肥大学).

Die Webseite mit dem Lehrmaterial dieses Kurses ist https://thomasweise.github.io/programmingWithPython (siehe auch den QR-Kode unten rechts). Dort können Sie das Kursbuch (in Englisch) und diese Slides finden. Das Repository mit den Beispielprogrammen in Python finden Sie unter https://github.com/thomasWeise/programmingWithPythonCode.

Outline

Einleitung

Einleitung

- Dieser Kurs lehrt das Programmieren mit der Programmiersprache Python.

Einleitung

- Dieser Kurs lehrt das Programmieren mit der Programmiersprache Python.
- Was bedeutet *Programmieren*?

Einleitung

- Dieser Kurs lehrt das Programmieren mit der Programmiersprache Python.
- Was bedeutet *Programmieren*?
- Programmieren bedeutet, dass wir Aufgaben an den Computer delegieren.

Einleitung

- Dieser Kurs lehrt das Programmieren mit der Programmiersprache Python.
- Was bedeutet *Programmieren*?
- Programmieren bedeutet, dass wir Aufgaben an den Computer delegieren.
- Wir haben eine Aufgabe zu erledigen, irgendeine Sache.

Einleitung

- Dieser Kurs lehrt das Programmieren mit der Programmiersprache Python.
- Was bedeutet *Programmieren*?
- Programmieren bedeutet, dass wir Aufgaben an den Computer delegieren.
- Wir haben eine Aufgabe zu erledigen, irgendeine Sache.
- Vielleicht ist sie zu kompliziert und dauert zulange.

Einleitung

- Dieser Kurs lehrt das Programmieren mit der Programmiersprache Python.
- Was bedeutet *Programmieren*?
- Programmieren bedeutet, dass wir Aufgaben an den Computer delegieren.
- Wir haben eine Aufgabe zu erledigen, irgendeine Sache.
- Vielleicht ist sie zu kompliziert und dauert zulange.
- Vielleicht ist es etwas, das wir sehr oft machen müssen.

Einleitung

- Dieser Kurs lehrt das Programmieren mit der Programmiersprache Python.
- Was bedeutet *Programmieren*?
- Programmieren bedeutet, dass wir Aufgaben an den Computer delegieren.
- Wir haben eine Aufgabe zu erledigen, irgendeine Sache.
- Vielleicht ist sie zu kompliziert und dauert zulange.
- Vielleicht ist es etwas, das wir sehr oft machen müssen.
- Vielleicht ist es etwas, das wir physisch nicht selbst machen können.

Einleitung

- Dieser Kurs lehrt das Programmieren mit der Programmiersprache Python.
- Was bedeutet *Programmieren*?
- Programmieren bedeutet, dass wir Aufgaben an den Computer delegieren.
- Wir haben eine Aufgabe zu erledigen, irgendeine Sache.
- Vielleicht ist sie zu kompliziert und dauert zulange.
- Vielleicht ist es etwas, das wir sehr oft machen müssen.
- Vielleicht ist es etwas, das wir physisch nicht selbst machen können.
- Vielleicht sind wir einfach faul.

Einleitung

- Dieser Kurs lehrt das Programmieren mit der Programmiersprache Python.
- Was bedeutet *Programmieren*?
- Programmieren bedeutet, dass wir Aufgaben an den Computer delegieren.
- Wir haben eine Aufgabe zu erledigen, irgendeine Sache.
- Vielleicht ist sie zu kompliziert und dauert zulange.
- Vielleicht ist es etwas, das wir sehr oft machen müssen.
- Vielleicht ist es etwas, das wir physisch nicht selbst machen können.
- Vielleicht sind wir einfach faul.
- Also wollen wir, dass der Computer es für uns macht.

Einleitung

- Wenn wir eine Aufgabe an eine andere Person delegieren, dann müssen wir die Aufgabe erklären.

Einleitung

- Wenn wir eine Aufgabe an eine andere Person delegieren, dann müssen wir die Aufgabe erklären.
- Wenn Sie der Chefkoch in einer Küche sind, dann müssen Sie dem Azubikoch erklären: „Erst musst Du die Kartoffeln waschen, dann schälen, und dann kannst Du sie kochen.“

Einleitung

- Wenn wir eine Aufgabe an eine andere Person delegieren, dann müssen wir die Aufgabe erklären.
- Wenn Sie der Chefkoch in einer Küche sind, dann müssen Sie dem Azubikoch erklären: „Erst musst Du die Kartoffeln waschen, dann schälen, und dann kannst Du sie kochen."
- Wenn Sie zum Friseur gehen um sich die Haare schön machen zu lassen, dann sagen Sie zum Beispiel: „Wasch meine Haare, schneide sie oben auf 1cm, trimm die Seiten, und färbe sie grün."

Einleitung

- Wenn wir eine Aufgabe an eine andere Person delegieren, dann müssen wir die Aufgabe erklären.
- Wenn Sie der Chefkoch in einer Küche sind, dann müssen Sie dem Azubikoch erklären: „Erst musst Du die Kartoffeln waschen, dann schälen, und dann kannst Du sie kochen."
- Wenn Sie zum Friseur gehen um sich die Haare schön machen zu lassen, dann sagen Sie zum Beispiel: „Wasch meine Haare, schneide sie oben auf 1cm, trimm die Seiten, und färbe sie grün."
- Wir geben der anderen Person eine klare und eindeutige Sequenz von Anweisungen in einer Sprache, die sie versteht.

Einleitung

- Wenn wir eine Aufgabe an eine andere Person delegieren, dann müssen wir die Aufgabe erklären.
- Wenn Sie der Chefkoch in einer Küche sind, dann müssen Sie dem Azubikoch erklären: „Erst musst Du die Kartoffeln waschen, dann schälen, und dann kannst Du sie kochen.“
- Wenn Sie zum Friseur gehen um sich die Haare schön machen zu lassen, dann sagen Sie zum Beispiel: „Wasch meine Haare, schneide sie oben auf 1cm, trimm die Seiten, und färbe sie grün.“
- Wir geben der anderen Person eine klare und eindeutige Sequenz von Anweisungen in einer Sprache, die sie versteht.
- In diesem Kurs lernen Sie, wie Sie das selbe mit einem Computer machen können.

Programmieren vs. Softwareentwicklung

Programmieren

Definition: Programm

Ein *Programm* ist eine eindeutige Sequenz von Berechnungsanweisungen für einen Computer um ein spezifisches Ziel zu erreichen.

Programmieren

Definition: Programm

Ein *Programm* ist eine eindeutige Sequenz von Berechnungsanweisungen für einen Computer um ein spezifisches Ziel zu erreichen.

Definition: Programmieren

Programmieren ist das Schreiben eines Programms[57].

Programmieren

- In der überwältigen Mehrheit der Fälle erstellen wir ein Programm *nicht*, um es nur ein einziges Mal auszuführen.

Programmieren

- In der überwältigen Mehrheit der Fälle erstellen wir ein Programm *nicht*, um es nur ein einziges Mal auszuführen.
- Wenn wir Aufgaben im realen Leben delegieren, ist das ja ganz ähnlich.

Programmieren

- In der überwältigen Mehrheit der Fälle erstellen wir ein Programm *nicht*, um es nur ein einziges Mal auszuführen.
- Wenn wir Aufgaben im realen Leben delegieren, ist das ja ganz ähnlich.
- Als Chefkoch „geben“ Sie das „Programm“ *Kartoffeln kochen* in den Azubikoch ja auch nur einmal „ein.“

Programmieren

- In der überwältigen Mehrheit der Fälle erstellen wir ein Programm *nicht*, um es nur ein einziges Mal auszuführen.
- Wenn wir Aufgaben im realen Leben delegieren, ist das ja ganz ähnlich.
- Als Chefkoch „geben“ Sie das „Programm“ *Kartoffeln kochen* in den Azubikoch ja auch nur einmal „ein.“
- Danach wollen Sie in der Lage sein, dieses „Programm“ auszuführen, in dem Sie sagen: „Koch doch bitte 2kg Kartoffeln.“

Programmieren

- In der überwältigen Mehrheit der Fälle erstellen wir ein Programm *nicht*, um es nur ein einziges Mal auszuführen.
- Wenn wir Aufgaben im realen Leben delegieren, ist das ja ganz ähnlich.
- Als Chefkoch „geben“ Sie das „Programm“ *Kartoffeln kochen* in den Azubikoch ja auch nur einmal „ein.“
- Danach wollen Sie in der Lage sein, dieses „Programm“ auszuführen, in dem Sie sagen: „Koch doch bitte 2kg Kartoffeln.“
- Solche „Programme“ haben also sogar oft implizite Parameter, wie zum Beispiel das eben erwähnte Gewicht von 2kg.

Programmieren

- In der überwältigen Mehrheit der Fälle erstellen wir ein Programm *nicht*, um es nur ein einziges Mal auszuführen.
- Wenn wir Aufgaben im realen Leben delegieren, ist das ja ganz ähnlich.
- Als Chefkoch „geben“ Sie das „Programm“ *Kartoffeln kochen* in den Azubikoch ja auch nur einmal „ein.“
- Danach wollen Sie in der Lage sein, dieses „Programm“ auszuführen, in dem Sie sagen: „Koch doch bitte 2kg Kartoffeln.“
- Solche „Programme“ haben also sogar oft implizite Parameter, wie zum Beispiel das eben erwähnte Gewicht von 2kg.
- Wenn Sie das nächste Mal zum Friseur gehen, wollen Sie vielleicht sagen können: „Das selbe wie immer, aber heute färbe sie blau.“

Programmieren

- In unseren täglichen Interaktionen erfolgt das Erstellen von wiederverwendbaren, parameterisierten Programmen sehr oft und sehr implizit.

Programmieren

- In unseren täglichen Interaktionen erfolgt das Erstellen von wiederverwendbaren, parameterisierten Programmen sehr oft und sehr implizit.
- Wir denken darüber selten explizit nach.

Programmieren

- In unseren täglichen Interaktionen erfolgt das Erstellen von wiederverwendbaren, parameterisierten Programmen sehr oft und sehr implizit.
- Wir denken darüber selten explizit nach.
- Wenn wir jedoch Computer programmieren, dann denken wir sehr wohl explizit darüber nach.

Programmieren

- In unseren täglichen Interaktionen erfolgt das Erstellen von wiederverwendbaren, parameterisierten Programmen sehr oft und sehr implizit.
- Wir denken darüber selten explizit nach.
- Wenn wir jedoch Computer programmieren, dann denken wir sehr wohl explizit darüber nach.
- Gleich von Anfang an.

Programmieren

- In unseren täglichen Interaktionen erfolgt das Erstellen von wiederverwendbaren, parameterisierten Programmen sehr oft und sehr implizit.
- Wir denken darüber selten explizit nach.
- Wenn wir jedoch Computer programmieren, dann denken wir sehr wohl explizit darüber nach.
- Gleich von Anfang an.
- Programmieren ist aber nur ein Teil von Softwareentwicklung.

Softwareentwicklung

- Später, in Ihrem Job, wollen Sie ein Programm entwickeln, das eine bestimmte Aufgabe löst.

Softwareentwicklung

- Später, in Ihrem Job, wollen Sie ein Programm entwickeln, das eine bestimmte Aufgabe löst.
  1. Sie schreiben das Program.

Softwareentwicklung

- Später, in Ihrem Job, wollen Sie ein Programm entwickeln, das eine bestimmte Aufgabe löst.
  1. Sie schreiben das Program.
  2. Dann haben Sie die Datei mit dem Programmkode.

Softwareentwicklung

- Später, in Ihrem Job, wollen Sie ein Programm entwickeln, das eine bestimmte Aufgabe löst.
  1. Sie schreiben das Program.
  2. Dann haben Sie die Datei mit dem Programmkode.
  3. Das Problem ist gelöst.

Softwareentwicklung

- Später, in Ihrem Job, wollen Sie ein Programm entwickeln, das eine bestimmte Aufgabe löst.
  1. Sie schreiben das Program.
  2. Dann haben Sie die Datei mit dem Programmkode.
  3. Das Problem ist gelöst.
- Ist das so einfach?

Softwareentwicklung

- Später, in Ihrem Job, wollen Sie ein Programm entwickeln, das eine bestimmte Aufgabe löst.
  1. Sie schreiben das Program.
  2. Dann haben Sie die Datei mit dem Programmkode.
  3. Das Problem ist gelöst.
- Ist das so einfach? Nein.

Softwareentwicklung

- Später, in Ihrem Job, wollen Sie ein Programm entwickeln, das eine bestimmte Aufgabe löst.
  1. Sie schreiben das Program.
  2. Dann haben Sie die Datei mit dem Programmkode.
  3. Das Problem ist gelöst.
- Ist das so einfach? Nein.
  1. Vielleicht fragen Sie sich, ob Sie einen Fehler gemacht haben.

Softwareentwicklung

- Später, in Ihrem Job, wollen Sie ein Programm entwickeln, das eine bestimmte Aufgabe löst.
  1. Sie schreiben das Program.
  2. Dann haben Sie die Datei mit dem Programmkode.
  3. Das Problem ist gelöst.
- Ist das so einfach? Nein.
  1. Vielleicht fragen Sie sich, ob Sie einen Fehler gemacht haben. Leute machen Fehler.

Softwareentwicklung

- Später, in Ihrem Job, wollen Sie ein Programm entwickeln, das eine bestimmte Aufgabe löst.
  1. Sie schreiben das Program.
  2. Dann haben Sie die Datei mit dem Programmkode.
  3. Das Problem ist gelöst.
- Ist das so einfach? Nein.
  1. Vielleicht fragen Sie sich, ob Sie einen Fehler gemacht haben. Leute machen Fehler. Je schwieriger die Aufgabe ist, die wir lösen wollen, je mehr Programmkode wir schreiben, desto wahrscheinlicher ist es, dass wir irgendwo irgendeinen Fehler machen.

Softwareentwicklung

- Später, in Ihrem Job, wollen Sie ein Programm entwickeln, das eine bestimmte Aufgabe löst.
  1. Sie schreiben das Program.
  2. Dann haben Sie die Datei mit dem Programmkode.
  3. Das Problem ist gelöst.
- Ist das so einfach? Nein.
  1. Vielleicht fragen Sie sich, ob Sie einen Fehler gemacht haben. Leute machen Fehler. Je schwieriger die Aufgabe ist, die wir lösen wollen, je mehr Programmkode wir schreiben, desto wahrscheinlicher ist es, dass wir irgendwo irgendeinen Fehler machen. Also müssen Sie Ihre Programme testen.

Softwareentwicklung

- Später, in Ihrem Job, wollen Sie ein Programm entwickeln, das eine bestimmte Aufgabe löst.
  1. Sie schreiben das Program.
  2. Dann haben Sie die Datei mit dem Programmkode.
  3. Das Problem ist gelöst.
- Ist das so einfach? Nein.
  1. Vielleicht fragen Sie sich, ob Sie einen Fehler gemacht haben. Leute machen Fehler. Je schwieriger die Aufgabe ist, die wir lösen wollen, je mehr Programmkode wir schreiben, desto wahrscheinlicher ist es, dass wir irgendwo irgendeinen Fehler machen. Also müssen Sie Ihre Programme testen.
  2. Was passiert wenn jemand anders Ihre Programme später verwenden will?

Softwareentwicklung

- Später, in Ihrem Job, wollen Sie ein Programm entwickeln, das eine bestimmte Aufgabe löst.
  1. Sie schreiben das Program.
  2. Dann haben Sie die Datei mit dem Programmkode.
  3. Das Problem ist gelöst.
- Ist das so einfach? Nein.
  1. Vielleicht fragen Sie sich, ob Sie einen Fehler gemacht haben. Leute machen Fehler. Je schwieriger die Aufgabe ist, die wir lösen wollen, je mehr Programmkode wir schreiben, desto wahrscheinlicher ist es, dass wir irgendwo irgendeinen Fehler machen. Also müssen Sie Ihre Programme testen.
  2. Was passiert wenn jemand anders Ihre Programme später verwenden will? Sie müssen eine klare Dokumentation schreiben.

Softwareentwicklung

- Später, in Ihrem Job, wollen Sie ein Programm entwickeln, das eine bestimmte Aufgabe löst.
  1. Sie schreiben das Program.
  2. Dann haben Sie die Datei mit dem Programmkode.
  3. Das Problem ist gelöst.
- Ist das so einfach? Nein.
  1. Vielleicht fragen Sie sich, ob Sie einen Fehler gemacht haben. Leute machen Fehler. Je schwieriger die Aufgabe ist, die wir lösen wollen, je mehr Programmkode wir schreiben, desto wahrscheinlicher ist es, dass wir irgendwo irgendeinen Fehler machen. Also müssen Sie Ihre Programme testen.
  2. Was passiert wenn jemand anders Ihre Programme später verwenden will? Sie müssen eine klare Dokumentation schreiben.
  3. Was, wenn Ihr Kode Funktionen zur Verfügung stellt, die andere verwenden können?

Softwareentwicklung

- Später, in Ihrem Job, wollen Sie ein Programm entwickeln, das eine bestimmte Aufgabe löst.
  1. Sie schreiben das Program.
  2. Dann haben Sie die Datei mit dem Programmkode.
  3. Das Problem ist gelöst.
- Ist das so einfach? Nein.
  1. Vielleicht fragen Sie sich, ob Sie einen Fehler gemacht haben. Leute machen Fehler. Je schwieriger die Aufgabe ist, die wir lösen wollen, je mehr Programmkode wir schreiben, desto wahrscheinlicher ist es, dass wir irgendwo irgendeinen Fehler machen. Also müssen Sie Ihre Programme testen.
  2. Was passiert wenn jemand anders Ihre Programme später verwenden will? Sie müssen eine klare Dokumentation schreiben.
  3. Was, wenn Ihr Kode Funktionen zur Verfügung stellt, die andere verwenden können? Die Ein- und Ausgabedatentypen müssen klar spezifiziert werden.

Softwareentwicklung

- Später, in Ihrem Job, wollen Sie ein Programm entwickeln, das eine bestimmte Aufgabe löst.
  1. Sie schreiben das Program.
  2. Dann haben Sie die Datei mit dem Programmkode.
  3. Das Problem ist gelöst.
- Ist das so einfach? Nein.
  1. Vielleicht fragen Sie sich, ob Sie einen Fehler gemacht haben. Leute machen Fehler. Je schwieriger die Aufgabe ist, die wir lösen wollen, je mehr Programmkode wir schreiben, desto wahrscheinlicher ist es, dass wir irgendwo irgendeinen Fehler machen. Also müssen Sie Ihre Programme testen.
  2. Was passiert wenn jemand anders Ihre Programme später verwenden will? Sie müssen eine klare Dokumentation schreiben.
  3. Was, wenn Ihr Kode Funktionen zur Verfügung stellt, die andere verwenden können? Die Ein- und Ausgabedatentypen müssen klar spezifiziert werden.
  4. Was, wenn jemand anders Ihren Kode lesen und damit arbeiten soll?

Softwareentwicklung

- Später, in Ihrem Job, wollen Sie ein Programm entwickeln, das eine bestimmte Aufgabe löst.
  1. Sie schreiben das Program.
  2. Dann haben Sie die Datei mit dem Programmkode.
  3. Das Problem ist gelöst.
- Ist das so einfach? Nein.
  1. Vielleicht fragen Sie sich, ob Sie einen Fehler gemacht haben. Leute machen Fehler. Je schwieriger die Aufgabe ist, die wir lösen wollen, je mehr Programmkode wir schreiben, desto wahrscheinlicher ist es, dass wir irgendwo irgendeinen Fehler machen. Also müssen Sie Ihre Programme testen.
  2. Was passiert wenn jemand anders Ihre Programme später verwenden will? Sie müssen eine klare Dokumentation schreiben.
  3. Was, wenn Ihr Kode Funktionen zur Verfügung stellt, die andere verwenden können? Die Ein- und Ausgabedatentypen müssen klar spezifiziert werden.
  4. Was, wenn jemand anders Ihren Kode lesen und damit arbeiten soll? Ihr Kode muss klar sein und konsistent einem ordentlichen Stil folgen[76].

Softwareentwicklung

- Später, in Ihrem Job, wollen Sie ein Programm entwickeln, das eine bestimmte Aufgabe löst.
  1. Sie schreiben das Program.
  2. Dann haben Sie die Datei mit dem Programmkode.
  3. Das Problem ist gelöst.
- Ist das so einfach? Nein.
  1. Vielleicht fragen Sie sich, ob Sie einen Fehler gemacht haben. Leute machen Fehler. Je schwieriger die Aufgabe ist, die wir lösen wollen, je mehr Programmkode wir schreiben, desto wahrscheinlicher ist es, dass wir irgendwo irgendeinen Fehler machen. Also müssen Sie Ihre Programme testen.
  2. Was passiert wenn jemand anders Ihre Programme später verwenden will? Sie müssen eine klare Dokumentation schreiben.
  3. Was, wenn Ihr Kode Funktionen zur Verfügung stellt, die andere verwenden können? Die Ein- und Ausgabedatentypen müssen klar spezifiziert werden.
  4. Was, wenn jemand anders Ihren Kode lesen und damit arbeiten soll? Ihr Kode muss klar sein und konsistent einem ordentlichen Stil folgen[76].
- Alle diese Dinge müssen beachtet werden!

Softwareentwicklung

- Softwareentwicklung ist also mehr als nur Programmieren..

Softwareentwicklung

- Softwareentwicklung ist also mehr als nur Programmieren..
- Die meisten Berufe sind ja mehr als nur die „Haupttätigkeit“, die man damit assoziiert

Softwareentwicklung

- Softwareentwicklung ist also mehr als nur Programmieren..
- Die meisten Berufe sind ja mehr als nur die „Haupttätigkeit“, die man damit assoziiert
  - Sagen wir, Sie gehen zum Arzt um sich behandeln zu lassen.

Softwareentwicklung

- Softwareentwicklung ist also mehr als nur Programmieren..
- Die meisten Berufe sind ja mehr als nur die „Haupttätigkeit“, die man damit assoziiert
  - Sagen wir, Sie gehen zum Arzt um sich behandeln zu lassen.
  - Dann *hoffen* Sie, dass dieser gut ausgebildet ist, die entsprechenden Operationen durchzuführen.

Softwareentwicklung

- Softwareentwicklung ist also mehr als nur Programmieren..
- Die meisten Berufe sind ja mehr als nur die „Haupttätigkeit“, die man damit assoziiert
  - Sagen wir, Sie gehen zum Arzt um sich behandeln zu lassen.
  - Dann *hoffen* Sie, dass dieser gut ausgebildet ist, die entsprechenden Operationen durchzuführen.
  - Aber Sie *erwarten absolut,* dass er sich die Hände vor der Operation wäscht.

Softwareentwicklung

- Softwareentwicklung ist also mehr als nur Programmieren..
- Die meisten Berufe sind ja mehr als nur die „Haupttätigkeit“, die man damit assoziiert
  - Sagen wir, Sie gehen zum Arzt um sich behandeln zu lassen.
  - Dann *hoffen* Sie, dass dieser gut ausgebildet ist, die entsprechenden Operationen durchzuführen.
  - Aber Sie *erwarten absolut,* dass er sich die Hände vor der Operation wäscht.
- Für Programmierer gilt das selbe!

Softwareentwicklung

- Softwareentwicklung ist also mehr als nur Programmieren..
- Die meisten Berufe sind ja mehr als nur die „Haupttätigkeit“, die man damit assoziiert
  - Sagen wir, Sie gehen zum Arzt um sich behandeln zu lassen.
  - Dann *hoffen* Sie, dass dieser gut ausgebildet ist, die entsprechenden Operationen durchzuführen.
  - Aber Sie *erwarten absolut,* dass er sich die Hände vor der Operation wäscht.
- Für Programmierer gilt das selbe!
  - Sagen wir, Ihr Chef will, dass Sie ein Programm schreiben.

Softwareentwicklung

- Softwareentwicklung ist also mehr als nur Programmieren..
- Die meisten Berufe sind ja mehr als nur die „Haupttätigkeit“, die man damit assoziiert
  - Sagen wir, Sie gehen zum Arzt um sich behandeln zu lassen.
  - Dann *hoffen* Sie, dass dieser gut ausgebildet ist, die entsprechenden Operationen durchzuführen.
  - Aber Sie *erwarten absolut,* dass er sich die Hände vor der Operation wäscht.
- Für Programmierer gilt das selbe!
  - Sagen wir, Ihr Chef will, dass Sie ein Programm schreiben.
  - Dann *hofft* er, dass Sie ein Programm schreiben, das gut funktioniert.

Softwareentwicklung

- Softwareentwicklung ist also mehr als nur Programmieren..
- Die meisten Berufe sind ja mehr als nur die „Haupttätigkeit“, die man damit assoziiert
  - Sagen wir, Sie gehen zum Arzt um sich behandeln zu lassen.
  - Dann *hoffen* Sie, dass dieser gut ausgebildet ist, die entsprechenden Operationen durchzuführen.
  - Aber Sie *erwarten absolut,* dass er sich die Hände vor der Operation wäscht.
- Für Programmierer gilt das selbe!
  - Sagen wir, Ihr Chef will, dass Sie ein Programm schreiben.
  - Dann *hofft* er, dass Sie ein Programm schreiben, das gut funktioniert.
  - Aber er *erwartet,* dass der Kode den Sie produzieren lesbar ist, das Sie ihn getestet haben, und dass sie ihn dokumentiert haben.

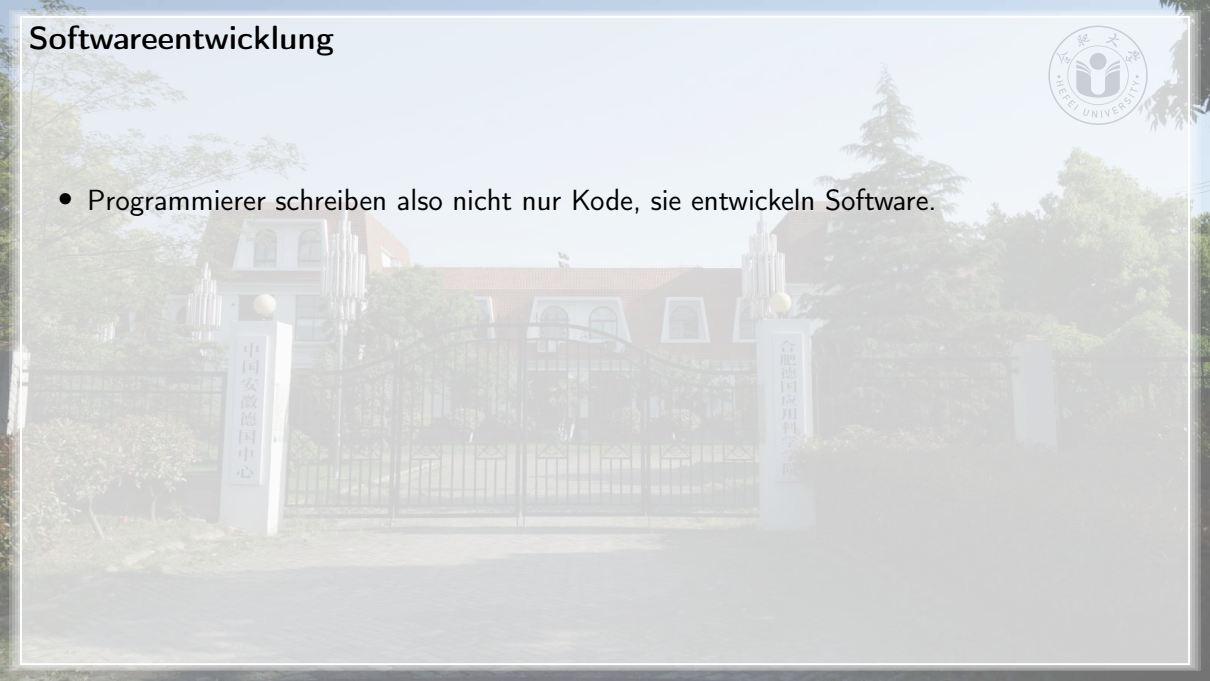
Softwareentwicklung

- Softwareentwicklung ist also mehr als nur Programmieren..
- Die meisten Berufe sind ja mehr als nur die „Haupttätigkeit“, die man damit assoziiert
  - Sagen wir, Sie gehen zum Arzt um sich behandeln zu lassen.
  - Dann *hoffen* Sie, dass dieser gut ausgebildet ist, die entsprechenden Operationen durchzuführen.
  - Aber Sie *erwarten absolut,* dass er sich die Hände vor der Operation wäscht.
- Für Programmierer gilt das selbe!
  - Sagen wir, Ihr Chef will, dass Sie ein Programm schreiben.
  - Dann *hofft* er, dass Sie ein Programm schreiben, das gut funktioniert.
  - Aber er *erwartet,* dass der Kode den Sie produzieren lesbar ist, das Sie ihn getestet haben, und dass sie ihn dokumentiert haben.
- Ich will nicht zu einem Arzt gehen, der sich nicht die Hände wäscht, bevor er mich operiert.

Softwareentwicklung

- Softwareentwicklung ist also mehr als nur Programmieren..
- Die meisten Berufe sind ja mehr als nur die „Haupttätigkeit“, die man damit assoziiert
  - Sagen wir, Sie gehen zum Arzt um sich behandeln zu lassen.
  - Dann *hoffen* Sie, dass dieser gut ausgebildet ist, die entsprechenden Operationen durchzuführen.
  - Aber Sie *erwarten absolut,* dass er sich die Hände vor der Operation wäscht.
- Für Programmierer gilt das selbe!
  - Sagen wir, Ihr Chef will, dass Sie ein Programm schreiben.
  - Dann *hofft* er, dass Sie ein Programm schreiben, das gut funktioniert.
  - Aber er *erwartet,* dass der Kode den Sie produzieren lesbar ist, das Sie ihn getestet haben, und dass sie ihn dokumentiert haben.
- Ich will nicht zu einem Arzt gehen, der sich nicht die Hände wäscht, bevor er mich operiert.
- Ich will Ihnen nicht Programmieren beibringen, ohne den Fokus auf *sauberen* Kode zu legen.

Softwareentwicklung



- Programmierer schreiben also nicht nur Kode, sie entwickeln Software.

Softwareentwicklung

- Programmierer schreiben also nicht nur Kode, sie entwickeln Software.
- Ein Großteil der Programmierer verbringt nur etwa 50% ihrer Zeit mit Programmieren[11,39].

Softwareentwicklung

- Programmierer schreiben also nicht nur Kode, sie entwickeln Software.
- Ein Großteil der Programmierer verbringt nur etwa 50% ihrer Zeit mit Programmieren[11,39].
- Andere Studien stellen sogar fest, dass weniger als 20% der Arbeitszeit mit Programmieren verbracht wird und vielleicht weitere 15% mit dem Korrigieren von Fehlern[45].

Softwareentwicklung

- Programmierer schreiben also nicht nur Kode, sie entwickeln Software.
- Ein Großteil der Programmierer verbringt nur etwa 50% ihrer Zeit mit Programmieren[11,39].
- Andere Studien stellen sogar fest, dass weniger als 20% der Arbeitszeit mit Programmieren verbracht wird und vielleicht weitere 15% mit dem Korrigieren von Fehlern[45].
- Natürlich fokussieren wir uns in diesem Kurs auf das Programmieren.

Softwareentwicklung

- Programmierer schreiben also nicht nur Kode, sie entwickeln Software.
- Ein Großteil der Programmierer verbringt nur etwa 50% ihrer Zeit mit Programmieren[11,39].
- Andere Studien stellen sogar fest, dass weniger als 20% der Arbeitszeit mit Programmieren verbracht wird und vielleicht weitere 15% mit dem Korrigieren von Fehlern[45].
- Natürlich fokussieren wir uns in diesem Kurs auf das Programmieren.
- Aber wir werden auch viele Dinge diskutieren, die darüber hinausgehen.

Softwareentwicklung

- Programmierer schreiben also nicht nur Kode, sie entwickeln Software.
- Ein Großteil der Programmierer verbringt nur etwa 50% ihrer Zeit mit Programmieren[11,39].
- Andere Studien stellen sogar fest, dass weniger als 20% der Arbeitszeit mit Programmieren verbracht wird und vielleicht weitere 15% mit dem Korrigieren von Fehlern[45].
- Natürlich fokussieren wir uns in diesem Kurs auf das Programmieren.
- Aber wir werden auch viele Dinge diskutieren, die darüber hinausgehen. Dinge, die in Ihren Werkzeuggürtel gehören.

Softwareentwicklung

- Programmierer schreiben also nicht nur Kode, sie entwickeln Software.
- Ein Großteil der Programmierer verbringt nur etwa 50% ihrer Zeit mit Programmieren[11,39].
- Andere Studien stellen sogar fest, dass weniger als 20% der Arbeitszeit mit Programmieren verbracht wird und vielleicht weitere 15% mit dem Korrigieren von Fehlern[45].
- Natürlich fokussieren wir uns in diesem Kurs auf das Programmieren.
- Aber wir werden auch viele Dinge diskutieren, die darüber hinausgehen. Dinge, die in Ihren Werkzeuggürtel gehören. Dinge, die Sie zu *guten* Programmierern machen.

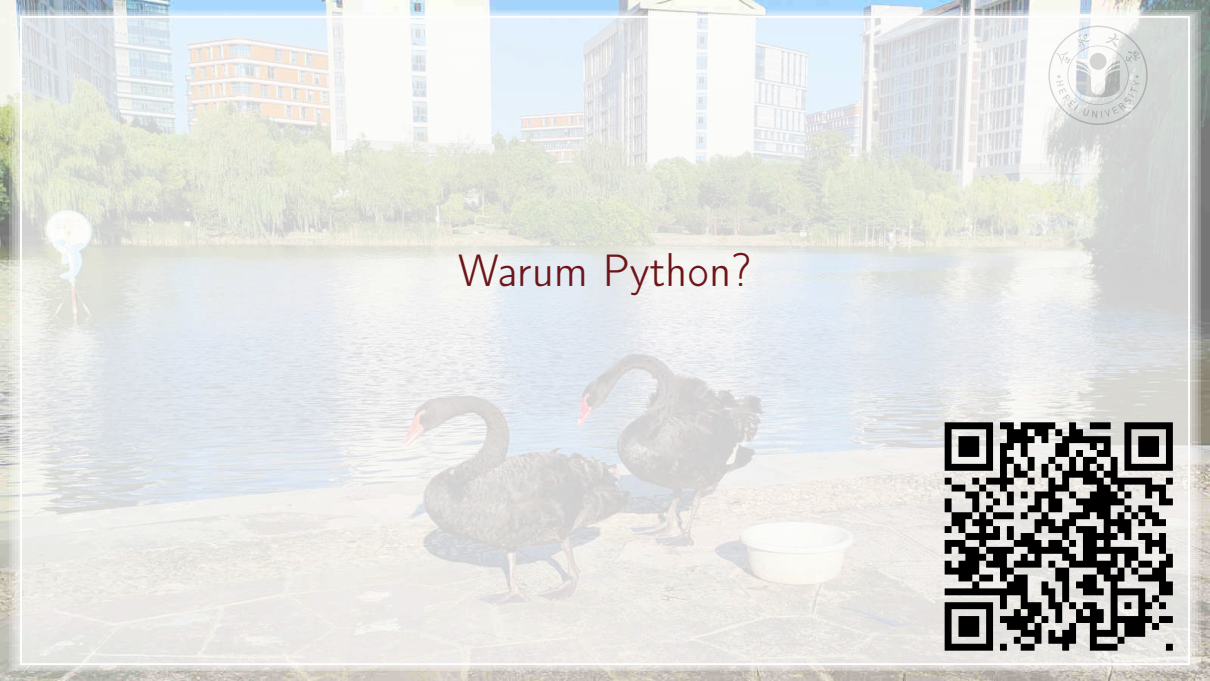
Softwareentwicklung

- Programmierer schreiben also nicht nur Kode, sie entwickeln Software.
- Ein Großteil der Programmierer verbringt nur etwa 50% ihrer Zeit mit Programmieren[11,39].
- Andere Studien stellen sogar fest, dass weniger als 20% der Arbeitszeit mit Programmieren verbracht wird und vielleicht weitere 15% mit dem Korrigieren von Fehlern[45].
- Natürlich fokussieren wir uns in diesem Kurs auf das Programmieren.
- Aber wir werden auch viele Dinge diskutieren, die darüber hinausgehen. Dinge, die in Ihren Werkzeuggürtel gehören. Dinge, die Sie zu *guten* Programmierern machen.
- Das Thema unserers Kurs ist das Entwickeln *guter* Software mit Python.

Softwareentwicklung

- Programmierer schreiben also nicht nur Kode, sie entwickeln Software.
- Ein Großteil der Programmierer verbringt nur etwa 50% ihrer Zeit mit Programmieren[11,39].
- Andere Studien stellen sogar fest, dass weniger als 20% der Arbeitszeit mit Programmieren verbracht wird und vielleicht weitere 15% mit dem Korrigieren von Fehlern[45].
- Natürlich fokussieren wir uns in diesem Kurs auf das Programmieren.
- Aber wir werden auch viele Dinge diskutieren, die darüber hinausgehen. Dinge, die in Ihren Werkzeuggürtel gehören. Dinge, die Sie zu *guten* Programmierern machen.
- Das Thema unserers Kurs ist das Entwickeln *guter* Software **mit Python**.

Warum Python?

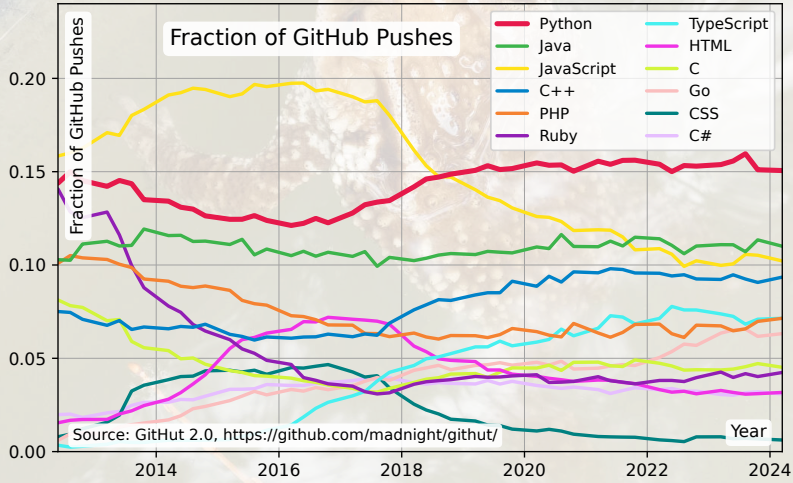


Warum Python?

1. Python ist eine sehr weitverbreitete Programmiersprache[6,8].

Warum Python?

1. Python ist eine sehr weitverbreitete Programmiersprache[6,8].



B2023G2GLS

Warum Python?

1. Python ist eine sehr weitverbreitete Programmiersprache[6,8]. Nach der jährlichen Stack Overflow Umfrage 2024[70], war Python die zweitpopulärste Programmiersprache nach JavaScript and HTML/CSS.

Warum Python?

1. Python ist eine sehr weitverbreitete Programmiersprache[6,8]. Nach der jährlichen Stack Overflow Umfrage 2024[70], war Python die zweitpopulärste Programmiersprache nach JavaScript and HTML/CSS. In GitHub's Octoverse Report vom Oktober 2024[18], war Python die populärste Programmiersprache (vor JavaScript).

Warum Python?

1. Python ist eine sehr weitverbreitete Programmiersprache[6,8]. Nach der jährlichen Stack Overflow Umfrage 2024[70], war Python die zweitpopulärste Programmiersprache nach JavaScript and HTML/CSS. In GitHub's Octoverse Report vom Oktober 2024[18], war Python die populärste Programmiersprache (vor JavaScript).
2. Python wird intensiv auf dem Gebiet der AI[64], ML[66], und DS[22] genutzt, die alle Zukunftstechnologien sind.

Warum Python?

1. Python ist eine sehr weitverbreitete Programmiersprache[6,8]. Nach der jährlichen Stack Overflow Umfrage 2024[70], war Python die zweitpopulärste Programmiersprache nach JavaScript and HTML/CSS. In GitHub's Octoverse Report vom Oktober 2024[18], war Python die populärste Programmiersprache (vor JavaScript).
2. Python wird intensiv auf dem Gebiet der AI[64], ML[66], und DS[22] genutzt, die alle Zukunftstechnologien sind.
3. Es existieren sehr mächtige Bibliotheken sowohl für Forschung als auch für die Produktentwicklung, z. B. NumPy[10,23,30], Pandas[3,37], Scikit-learn[54,59], SciPy[30,77], TensorFlow[1,34], PyTorch[53,59], Matplotlib[27,30,50], SimPy[82], und moptipy[80], um nur ein paar zu nennen.

Warum Python?

1. Python ist eine sehr weitverbreitete Programmiersprache[6,8]. Nach der jährlichen Stack Overflow Umfrage 2024[70], war Python die zweitpopulärste Programmiersprache nach JavaScript and HTML/CSS. In GitHub's Octoverse Report vom Oktober 2024[18], war Python die populärste Programmiersprache (vor JavaScript).
2. Python wird intensiv auf dem Gebiet der AI[64], ML[66], und DS[22] genutzt, die alle Zukunftstechnologien sind.
3. Es existieren sehr mächtige Bibliotheken sowohl für Forschung als auch für die Produktentwicklung, z. B. NumPy[10,23,30], Pandas[3,37], Scikit-learn[54,59], SciPy[30,77], TensorFlow[1,34], PyTorch[53,59], Matplotlib[27,30,50], SimPy[82], und moptipy[80], um nur ein paar zu nennen.
4. Python ist sehr einfach zu erlernen[21,75].

Warum Python?

1. Python ist eine sehr weitverbreitete Programmiersprache[6,8]. Nach der jährlichen Stack Overflow Umfrage 2024[70], war Python die zweitpopulärste Programmiersprache nach JavaScript and HTML/CSS. In GitHub's Octoverse Report vom Oktober 2024[18], war Python die populärste Programmiersprache (vor JavaScript).
2. Python wird intensiv auf dem Gebiet der AI[64], ML[66], und DS[22] genutzt, die alle Zukunftstechnologien sind.
3. Es existieren sehr mächtige Bibliotheken sowohl für Forschung als auch für die Produktentwicklung, z. B. NumPy[10,23,30], Pandas[3,37], Scikit-learn[54,59], SciPy[30,77], TensorFlow[1,34], PyTorch[53,59], Matplotlib[27,30,50], SimPy[82], und moptipy[80], um nur ein paar zu nennen.
4. Python ist sehr einfach zu erlernen[21,75]. Es hat eine einfache und saubere Syntax, die zu einer gut lesbaren Programmstruktur führt.

Warum Python?

1. Python ist eine sehr weitverbreitete Programmiersprache[6,8]. Nach der jährlichen Stack Overflow Umfrage 2024[70], war Python die zweitpopulärste Programmiersprache nach JavaScript and HTML/CSS. In GitHub's Octoverse Report vom Oktober 2024[18], war Python die populärste Programmiersprache (vor JavaScript).
2. Python wird intensiv auf dem Gebiet der AI[64], ML[66], und DS[22] genutzt, die alle Zukunftstechnologien sind.
3. Es existieren sehr mächtige Bibliotheken sowohl für Forschung als auch für die Produktentwicklung, z. B. NumPy[10,23,30], Pandas[3,37], Scikit-learn[54,59], SciPy[30,77], TensorFlow[1,34], PyTorch[53,59], Matplotlib[27,30,50], SimPy[82], und moptipy[80], um nur ein paar zu nennen.
4. Python ist sehr einfach zu erlernen[21,75]. Es hat eine einfache und saubere Syntax, die zu einer gut lesbaren Programmstruktur führt. Python hat auch sehr mächtige Standarddatentypen, wie z. B. Listen, Tuples, und Dictionaries.

Warum Python?

1. Python ist eine sehr weitverbreitete Programmiersprache[6,8]. Nach der jährlichen Stack Overflow Umfrage 2024[70], war Python die zweitpopulärste Programmiersprache nach JavaScript and HTML/CSS. In GitHub's Octoverse Report vom Oktober 2024[18], war Python die populärste Programmiersprache (vor JavaScript).
2. Python wird intensiv auf dem Gebiet der AI[64], ML[66], und DS[22] genutzt, die alle Zukunftstechnologien sind.
3. Es existieren sehr mächtige Bibliotheken sowohl für Forschung als auch für die Produktentwicklung, z. B. NumPy[10,23,30], Pandas[3,37], Scikit-learn[54,59], SciPy[30,77], TensorFlow[1,34], PyTorch[53,59], Matplotlib[27,30,50], SimPy[82], und moptipy[80], um nur ein paar zu nennen.
4. Python ist sehr einfach zu erlernen[21,75]. Es hat eine einfache und saubere Syntax, die zu einer gut lesbaren Programmstruktur führt. Python hat auch sehr mächtige Standarddatentypen, wie z. B. Listen, Tuples, und Dictionaries. Darum war Python auch die populärste Programmiersprache für Leute, die das Programmieren lernen wollen, nach der oben erwähnten Umfrage[70].

Python ist ein interpretierte Sprache

- Die meisten Programmiersprachen erfordern, dass Quellkode kompiliert wird.

Python ist ein interpretierte Sprache

- Die meisten Programmiersprachen erfordern, dass Quellkode kompiliert wird.

Programmieren in einer kompilierten Sprache wie C

Python ist ein interpretierte Sprache

- Die meisten Programmiersprachen erfordern, dass Quellkode kompiliert wird.

Programmieren in einer kompilierten Sprache wie C

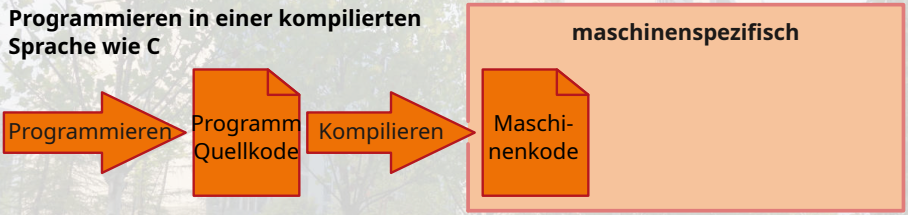
Python ist ein interpretierte Sprache

- Die meisten Programmiersprachen erfordern, dass Quellkode kompiliert wird.

Programmieren in einer kompilierten Sprache wie C

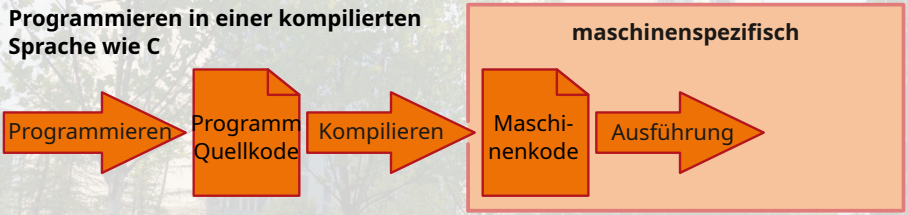
Python ist ein interpretierte Sprache

- Die meisten Programmiersprachen erfordern, dass Quellkode kompiliert wird.



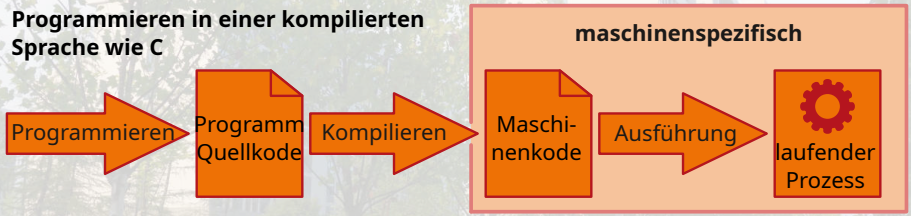
Python ist ein interpretierte Sprache

- Die meisten Programmiersprachen erfordern, dass Quellkode kompiliert wird.



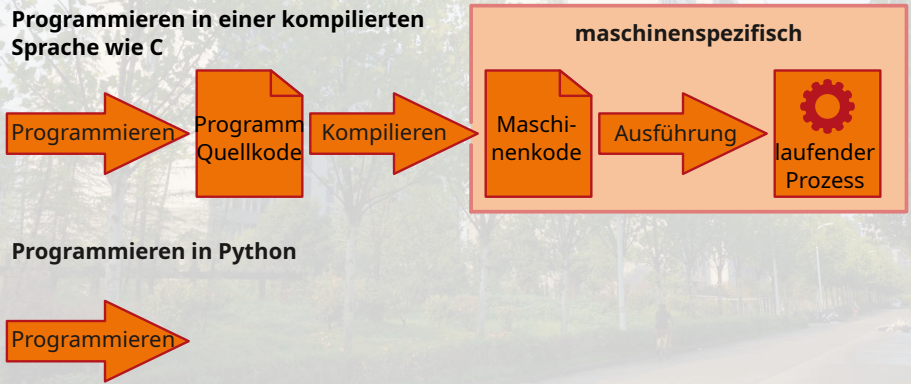
Python ist ein interpretierte Sprache

- Die meisten Programmiersprachen erfordern, dass Quellkode kompiliert wird.



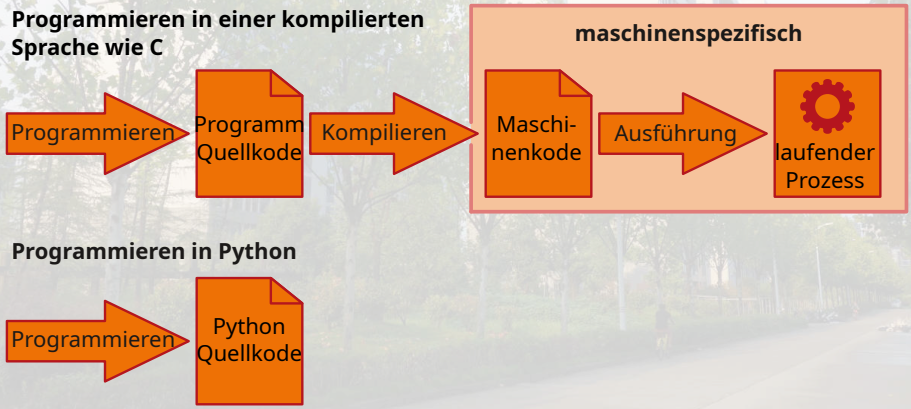
Python ist ein interpretierte Sprache

- Die meisten Programmiersprachen erfordern, dass Quellkode kompiliert wird.
- Python ist interpretiert.



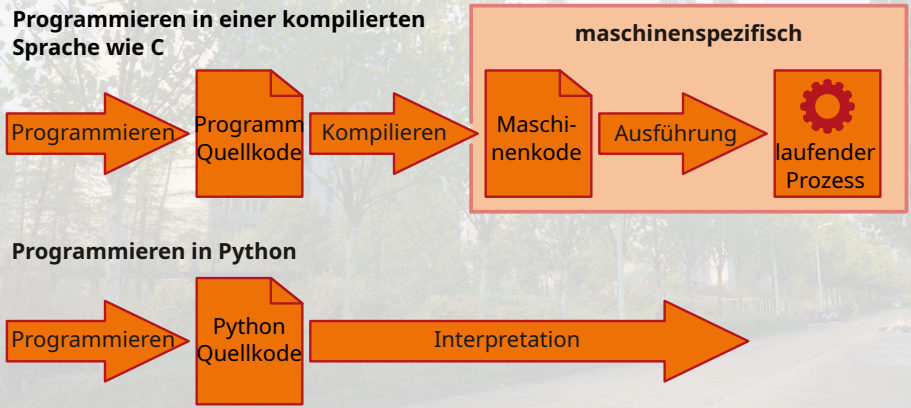
Python ist ein interpretierte Sprache

- Die meisten Programmiersprachen erfordern, dass Quellkode kompiliert wird.
- Python ist interpretiert.



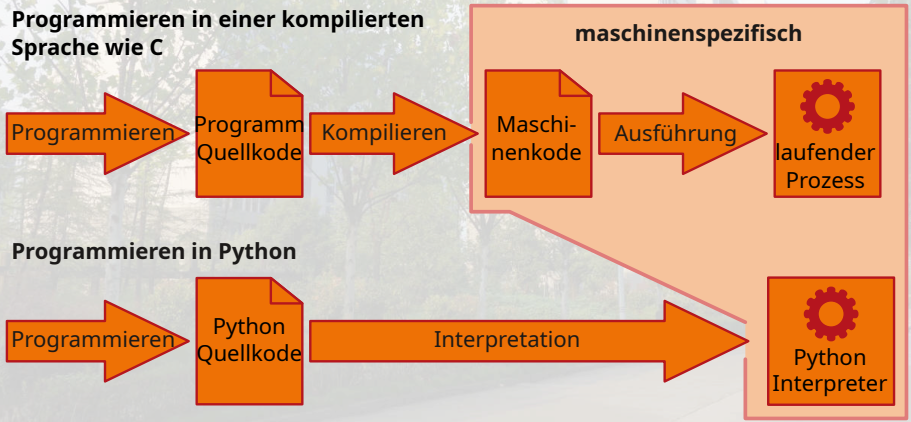
Python ist ein interpretierte Sprache

- Die meisten Programmiersprachen erfordern, dass Quellkode kompiliert wird.
- Python ist interpretiert.



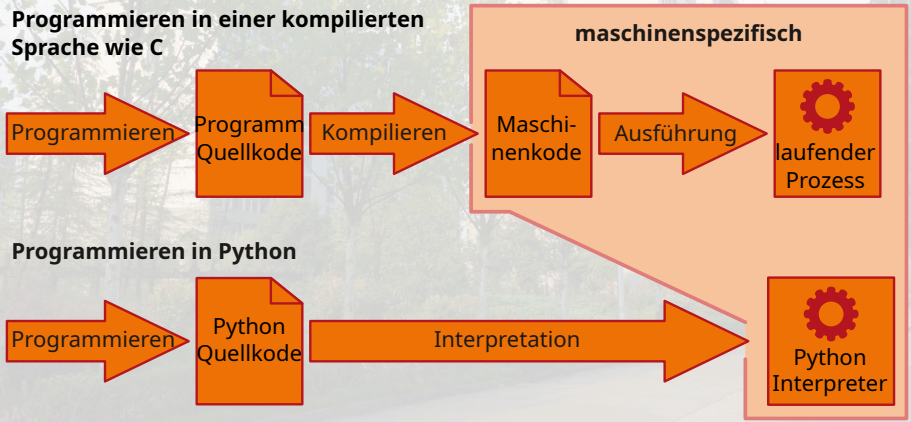
Python ist ein interpretierte Sprache

- Die meisten Programmiersprachen erfordern, dass Quellkode kompiliert wird.
- Python ist interpretiert.



Python ist ein interpretierte Sprache

- Die meisten Programmiersprachen erfordern, dass Quellkode kompiliert wird.
- Python ist interpretiert.
- Daher gibt es weniger Schritte im Buildprozess.



Literatur

Literatur

- Für diesen Kurs reicht unser Kursbuch[79].

Literatur

- Für diesen Kurs reicht unser Kursbuch[79].
- Weitere Bücher zu Python:
  - Bernd Klein. *Einführung in Python 3 – Für Ein- und Umsteiger*. 3., überarbeitete. Carl Hanser Verlag GmbH & Co. KG, 2018. ISBN: 978-3-446-45208-4. doi:10.3139/9783446453876,
  - Mark Lutz. *Learning Python*. 6. Aufl. O'Reilly Media, Inc., März 2025. ISBN: 978-1-0981-7130-8,
  - Kevin Wilson. *Python Made Easy*. Packt Publishing Ltd, Aug. 2024. ISBN: 978-1-83664-615-0,
  - Brett Slatkin. *Effective Python: 125 Specific Ways to Write Better Python*. 3. Aufl. Addison-Wesley Professional, Nov. 2024. ISBN: 978-0-13-817239-8,
  - John Hunt. *A Beginners Guide to Python 3 Programming*. 2. Aufl. Springer, 2023. ISBN: 978-3-031-35121-1. doi:10.1007/978-3-031-35122-8,
  - Eric Matthes. *Python Crash Course*. 3. Aufl. No Starch Press, Jan. 2023. ISBN: 978-1-7185-0270-3,
  - Paul Barry. *Head First Python*. 3. Aufl. O'Reilly Media, Inc., Aug. 2023. ISBN: 978-1-4920-5129-9,
  - Mike James. *Programmer's Python: Everything is an Object – Something Completely Different*. 2. Aufl. I/O Press, 25. Juni 2022.

Literatur

- Für diesen Kurs reicht unser Kursbuch[79].
- Weitere Bücher zu Softwaretests und kontinuierliche Integration:
  - Brian Okken. *Python Testing with pytest*. Pragmatic Bookshelf by The Pragmatic Programmers, L.L.C., Feb. 2022. ISBN: 978-1-68050-860-4,
  - Ashwin Pajankar. *Python Unit Test Automation: Automate, Organize, and Execute Unit Tests in Python*. Apress Media, LLC, Dez. 2021. ISBN: 978-1-4842-7854-3,
  - Alfredo Deza und Noah Gift. *Testing In Python*. Pragmatic AI Labs, Feb. 2020. ISBN: 979-8-6169-6064-1,
  - Moritz Lenz. *Python Continuous Integration and Delivery: A Concise Guide with Examples*. Apress Media, LLC, Dez. 2018. ISBN: 978-1-4842-4281-0,
  - Kristian Rother. *Pro Python Best Practices: Debugging, Testing and Maintenance*. Apress Media, LLC, März 2017. ISBN: 978-1-4842-2241-6.

Literatur

- Für diesen Kurs reicht unser Kursbuch[79].
- Weitere Bücher zu Data Science (DS), numerische Berechnungen, Visualisierung, und AI:
  - Wes McKinney. *Python for Data Analysis*. 3. Aufl. O'Reilly Media, Inc., Aug. 2022. ISBN: 978-1-0981-0403-0,
  - Ashwin Pajankar. *Hands-on Matplotlib: Learn Plotting and Visualizations with Python 3*. Apress Media, LLC, Nov. 2021. ISBN: 978-1-4842-7410-1,
  - Joel Grus. *Data Science from Scratch: First Principles with Python*. 2. Aufl. O'Reilly Media, Inc., Mai 2019. ISBN: 978-1-4920-4113-9,
  - Robert Johansson. *Numerical Python: Scientific Computing and Data Science Applications with NumPy, SciPy and Matplotlib*. Apress Media, LLC, Dez. 2018. ISBN: 978-1-4842-4246-9.

Literatur

- Für diesen Kurs reicht unser Kursbuch[79].
- Bücher zu weiteren Themen:
  - Aaron Maxwell. *What are f-strings in Python and how can I use them?* Infinite Skills Inc, Juni 2017. ISBN: 978-1-4919-9486-3,
  - Abhishek Ratan, Eric Chou, Pradeeban Kathiravelu und Dr. M.O. Faruque Sarker. *Python Network Programming*. Packt Publishing Ltd, Jan. 2019. ISBN: 978-1-78883-546-6.

Literatur

- Für diesen Kurs reicht unser Kursbuch[79].
- Onlineressourcen:
  - *Python 3 Documentation*. Python Software Foundation (PSF), 2001–2025. URL: https://docs.python.org/3,
  - The PEP Editors. *Index of Python Enhancement Proposals (PEPs)*. Python Enhancement Proposal (PEP) 0. Python Software Foundation (PSF), 13. Juli 2000. URL: https://peps.python.org,
  - *Real Python Tutorials*. DevCademy Media Inc., 2021–2025. URL: https://realpython.com,
  - *W3Schools: Python Tutorials*. Refsnes Data AS, 1999–2025. URL: https://www.w3schools.com/python,
  - Trey Hunner. *Python Morsels*. Python Morsels, 2025. URL: https://www.pythonmorsels.com,
  - Florian Bruhin. *Python f-Strings*. Bruhin Software, 31. Mai 2023. URL: https://fstring.help,
  - Benjamin Dicken. *PyFlo – The Beginners Guide to Becoming a Python Programmer*. 2023. URL: https://pyflo.net.

Zusammenfassung

Zusammenfassung

- Programmieren ist das Schreiben von Quellkode für Computerprogramme.

Zusammenfassung

- Programmieren ist das Schreiben von Quellkode für Computerprogramme.
- Wir können dafür eine Programmiersprache wie Python verwenden.

Zusammenfassung

- Programmieren ist das Schreiben von Quellkode für Computerprogramme.
- Wir können dafür eine Programmiersprache wie Python verwenden.
- Um gute, nützliche, und wartbare Programme zu schreiben, ist es nicht genug, eine Programmiersprache zu erlernen.

Zusammenfassung

- Programmieren ist das Schreiben von Quellkode für Computerprogramme.
- Wir können dafür eine Programmiersprache wie Python verwenden.
- Um gute, nützliche, und wartbare Programme zu schreiben, ist es nicht genug, eine Programmiersprache zu erlernen.
- Man muss auch die dazugehörigen Werkzeuge verstehen, die bewährten Praktiken, die Stilrichtlinien, man muss wissen, wie man Programme testet, wie man sie dokumentiert, und so weiter.

Zusammenfassung

- Programmieren ist das Schreiben von Quellkode für Computerprogramme.
- Wir können dafür eine Programmiersprache wie Python verwenden.
- Um gute, nützliche, und wartbare Programme zu schreiben, ist es nicht genug, eine Programmiersprache zu erlernen.
- Man muss auch die dazugehörigen Werkzeuge verstehen, die bewährten Praktiken, die Stilrichtlinien, man muss wissen, wie man Programme testet, wie man sie dokumentiert, und so weiter.
- Man muss die wichtigsten Komponenten der *Softwareentwicklung* verstehen.

Zusammenfassung

- Programmieren ist das Schreiben von Quellkode für Computerprogramme.
- Wir können dafür eine Programmiersprache wie Python verwenden.
- Um gute, nützliche, und wartbare Programme zu schreiben, ist es nicht genug, eine Programmiersprache zu erlernen.
- Man muss auch die dazugehörigen Werkzeuge verstehen, die bewährten Praktiken, die Stilrichtlinien, man muss wissen, wie man Programme testet, wie man sie dokumentiert, und so weiter.
- Man muss die wichtigsten Komponenten der *Softwareentwicklung* verstehen.
- Ich werde versuchen, Ihnen Programmieren zusammen mit mehrerer solcher Aspekte beizubringen.

Zusammenfassung

- Programmieren ist das Schreiben von Quellkode für Computerprogramme.
- Wir können dafür eine Programmiersprache wie Python verwenden.
- Um gute, nützliche, und wartbare Programme zu schreiben, ist es nicht genug, eine Programmiersprache zu erlernen.
- Man muss auch die dazugehörigen Werkzeuge verstehen, die bewährten Praktiken, die Stilrichtlinien, man muss wissen, wie man Programme testet, wie man sie dokumentiert, und so weiter.
- Man muss die wichtigsten Komponenten der *Softwareentwicklung* verstehen.
- Ich werde versuchen, Ihnen Programmieren zusammen mit mehrerer solcher Aspekte beizubringen.
- Wir nutzen die Programmiersprache Python

Zusammenfassung

- Programmieren ist das Schreiben von Quellkode für Computerprogramme.
- Wir können dafür eine Programmiersprache wie Python verwenden.
- Um gute, nützliche, und wartbare Programme zu schreiben, ist es nicht genug, eine Programmiersprache zu erlernen.
- Man muss auch die dazugehörigen Werkzeuge verstehen, die bewährten Praktiken, die Stilrichtlinien, man muss wissen, wie man Programme testet, wie man sie dokumentiert, und so weiter.
- Man muss die wichtigsten Komponenten der *Softwareentwicklung* verstehen.
- Ich werde versuchen, Ihnen Programmieren zusammen mit mehrerer solcher Aspekte beizubringen.
- Wir nutzen die Programmiersprache Python, weil sie einfach zu lernen ist

Zusammenfassung

- Programmieren ist das Schreiben von Quellkode für Computerprogramme.
- Wir können dafür eine Programmiersprache wie Python verwenden.
- Um gute, nützliche, und wartbare Programme zu schreiben, ist es nicht genug, eine Programmiersprache zu erlernen.
- Man muss auch die dazugehörigen Werkzeuge verstehen, die bewährten Praktiken, die Stilrichtlinien, man muss wissen, wie man Programme testet, wie man sie dokumentiert, und so weiter.
- Man muss die wichtigsten Komponenten der *Softwareentwicklung* verstehen.
- Ich werde versuchen, Ihnen Programmieren zusammen mit mehrerer solcher Aspekte beizubringen.
- Wir nutzen die Programmiersprache Python, weil sie einfach zu lernen ist, weit-verbreitet

Zusammenfassung

- Programmieren ist das Schreiben von Quellkode für Computerprogramme.
- Wir können dafür eine Programmiersprache wie Python verwenden.
- Um gute, nützliche, und wartbare Programme zu schreiben, ist es nicht genug, eine Programmiersprache zu erlernen.
- Man muss auch die dazugehörigen Werkzeuge verstehen, die bewährten Praktiken, die Stilrichtlinien, man muss wissen, wie man Programme testet, wie man sie dokumentiert, und so weiter.
- Man muss die wichtigsten Komponenten der *Softwareentwicklung* verstehen.
- Ich werde versuchen, Ihnen Programmieren zusammen mit mehrerer solcher Aspekte beizubringen.
- Wir nutzen die Programmiersprache Python, weil sie einfach zu lernen ist, weit-verbreitet, ein reiches Ökosystem von nützlichen Bibliotheken und Werkzeugen bietet

Zusammenfassung

- Programmieren ist das Schreiben von Quellkode für Computerprogramme.
- Wir können dafür eine Programmiersprache wie Python verwenden.
- Um gute, nützliche, und wartbare Programme zu schreiben, ist es nicht genug, eine Programmiersprache zu erlernen.
- Man muss auch die dazugehörigen Werkzeuge verstehen, die bewährten Praktiken, die Stilrichtlinien, man muss wissen, wie man Programme testet, wie man sie dokumentiert, und so weiter.
- Man muss die wichtigsten Komponenten der *Softwareentwicklung* verstehen.
- Ich werde versuchen, Ihnen Programmieren zusammen mit mehrerer solcher Aspekte beizubringen.
- Wir nutzen die Programmiersprache Python, weil sie einfach zu lernen ist, weit-verbreitet, ein reiches Ökosystem von nützlichen Bibliotheken und Werkzeugen bietet, und einen einfachen Buildprozess hat.

Zusammenfassung

- Programmieren ist das Schreiben von Quellkode für Computerprogramme.
- Wir können dafür eine Programmiersprache wie Python verwenden.
- Um gute, nützliche, und wartbare Programme zu schreiben, ist es nicht genug, eine Programmiersprache zu erlernen.
- Man muss auch die dazugehörigen Werkzeuge verstehen, die bewährten Praktiken, die Stilrichtlinien, man muss wissen, wie man Programme testet, wie man sie dokumentiert, und so weiter.
- Man muss die wichtigsten Komponenten der *Softwareentwicklung* verstehen.
- Ich werde versuchen, Ihnen Programmieren zusammen mit mehrerer solcher Aspekte beizubringen.
- Wir nutzen die Programmiersprache Python, weil sie einfach zu lernen ist, weit-verbreitet, ein reiches Ökosystem von nützlichen Bibliotheken und Werkzeugen bietet, und einen einfachen Buildprozess hat.
- Aber genug der Einleitung. Jetzt installieren wir erst mal Python!

谢谢您们！

Thank you!

Vielen Dank!

References I

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek Gordon Murray, Benoit Steiner, Paul A. Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu und Xiaoqiang Zheng. "TensorFlow: A System for Large-Scale Machine Learning". In: *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI'2016)*. 2.–4. Nov. 2016, Savannah, GA, USA. Hrsg. von Kimberly Keeton und Timothy Roscoe. Berkeley, CA, USA: USENIX Association, 2016, S. 265–283. ISBN: 978-1-931971-33-1. URL: https://www.usenix.org/conference/osdi16/technical-sessions/presentation/abadi (besucht am 2024-06-26) (siehe S. 71–80, 126).

[2] Paul Barry. *Head First Python*. 3. Aufl. Sebastopol, CA, USA: O'Reilly Media, Inc., Aug. 2023. ISBN: 978-1-4920-5129-9 (siehe S. 94, 95).

[3] David M. Beazley. "Data Processing with Pandas". *;login: Usenix Magazin* 37(6), Dez. 2012. Berkeley, CA, USA: USENIX Association. ISSN: 1044-6397. URL: https://www.usenix.org/publications/login/december-2012-volume-37-number-6/data-processing-pandas (besucht am 2024-06-25) (siehe S. 71–80, 125).

[4] Kent L. Beck. *JUnit Pocket Guide*. Sebastopol, CA, USA: O'Reilly Media, Inc., Sep. 2004. ISBN: 978-0-596-00743-0 (siehe S. 126).

[5] Tim Berners-Lee. *Re: Qualifiers on Hypertext links...* Geneva, Switzerland: World Wide Web project, European Organization for Nuclear Research (CERN) und Newsgroups: alt.hypertext, 6. Aug. 1991. URL: https://www.w3.org/People/Berners-Lee/1991/08/art-6484.txt (besucht am 2025-02-05) (siehe S. 124, 126).

[6] Fabian Beuke. *GitHut 2.0: GitHub Language Statistics*. San Francisco, CA, USA: GitHub Inc, 2023. URL: https://madnight.github.io/githut (besucht am 2024-06-24) (siehe S. 71–80).

[7] Florian Bruhin. *Python f-Strings*. Winterthur, Switzerland: Bruhin Software, 31. Mai 2023. URL: https://fstring.help (besucht am 2024-07-25) (siehe S. 99, 124).

[8] Oscar Castro, Pierrick Bruneau, Jean-Sébastien Sottet und Dario Torregrossa. "Landscape of High-Performance Python to Develop Data Science and Machine Learning Applications". *ACM Computing Surveys (CSUR)* 56(3):65:1–65:30, 2024. New York, NY, USA: Association for Computing Machinery (ACM). ISSN: 0360-0300. doi:10.1145/3617588 (siehe S. 71–80).

[9] Paul Deitel, Harvey Deitel und Abbey Deitel. *Internet & World Wide WebW[: How to Program*. 5. Aufl. Hoboken, NJ, USA: Pearson Education, Inc., Nov. 2011. ISBN: 978-0-13-299045-5 (siehe S. 126).

References II

[10] Justin Dennison, Cherokee Boose und Peter van Rysdam. *Intro to NumPy*. Centennial, CO, USA: ACI Learning. Birmingham, England, UK: Packt Publishing Ltd, Juni 2024. ISBN: 978-1-83620-863-1 (siehe S. 71–80, 125).

[11] *Developer Survey 2019: Open Source Runtime Pains*. Vancouver, BC, Canada: ActiveState Software Inc., 30. Apr. 2019. URL: https://cdn.activestate.com/wp-content/uploads/2019/05/ActiveState-Developer-Survey-2019-Open-Source-Runtime-Pains.pdf (besucht am 2024-12-29) (siehe S. 61–69).

[12] Alfredo Deza und Noah Gift. *Testing In Python*. San Francisco, CA, USA: Pragmatic AI Labs, Feb. 2020. ISBN: 979-8-6169-6064-1 (siehe S. 96, 125).

[13] Benjamin Dicken. *PyFlo – The Beginners Guide to Becoming a Python Programmer*. 2023. URL: https://pyflo.net (besucht am 2025-08-28) (siehe S. 99).

[14] Slobodan Dmitrović. *Modern C for Absolute Beginners: A Friendly Introduction to the C Programming Language*. New York, NY, USA: Apress Media, LLC, März 2024. ISBN: 979-8-8688-0224-9 (siehe S. 124).

[15] *ECMAScript Language Specification*. Standard ECMA-262, 3rd Edition. Geneva, Switzerland: Ecma International, Dez. 1999. URL: https://ecma-international.org/wp-content/uploads/ECMA-262_3rd_edition_december_1999.pdf (besucht am 2024-12-15) (siehe S. 124).

[16] "Formatted String Literals". In: *Python 3 Documentation. The Python Tutorial*. Beaverton, OR, USA: Python Software Foundation (PSF), 2001–2025. Kap. 7.1.1. URL: https://docs.python.org/3/tutorial/inputoutput.html#formatted-string-literals (besucht am 2024-07-25) (siehe S. 124).

[17] Bhavesh Gawade. "Mastering F-Strings in Python: Efficient String Handling in Python Using Smart F-Strings". In: *C O D E B*. Mumbai, Maharashtra, India: Code B Solutions Pvt Ltd, 25. Apr.–3. Juni 2025. URL: https://code-b.dev/blog/f-strings-in-python (besucht am 2025-08-04) (siehe S. 124).

[18] GitHub Staff. *Octoverse: AI leads Python to top language as the number of global developers surges*. San Francisco, CA, USA: GitHub Inc, 29. Okt. 2024. URL: https://github.blog/news-insights/octoverse/octoverse-2024 (besucht am 2025-01-07) (siehe S. 71–80).

[19] Michael Goodwin. *What is an API?* Armonk, NY, USA: International Business Machines Corporation (IBM), 9. Apr. 2024. URL: https://www.ibm.com/topics/api (besucht am 2024-12-12) (siehe S. 124).

References III

[20] Olaf Górski. "Why f-strings are awesome: Performance of different string concatenation methods in Python". In: *DEV Community*. Sacramento, CA, USA: DEV Community Inc., 8. Nov. 2022. URL: https://dev.to/grski/performance-of-different-string-concatenation-methods-in-python-why-f-strings-are-awesome-2e97 (besucht am 2025-08-04) (siehe S. 124).

[21] Linda Grandell, Mia Peltomäki, Ralph-Johan Back und Tapio Salakoski. "Why complicate things? Introducing Programming in High School using Python". In: *8th Australasian Conference on Computing Education (ACE'2006)*. 16.–19. Jan. 2006, Hobart, TAS, Australia. Hrsg. von Denise Tolhurst und Samuel Mann. Bd. 52. New York, NY, USA: Association for Computing Machinery (ACM), 2006, S. 71–80. ISBN: 978-1-920682-34-7. doi:10.5555/1151869.1151880 (siehe S. 71–80).

[22] Joel Grus. *Data Science from Scratch: First Principles with Python*. 2. Aufl. Sebastopol, CA, USA: O'Reilly Media, Inc., Mai 2019. ISBN: 978-1-4920-4113-9 (siehe S. 71–80, 97, 124).

[23] Charles R. Harris, K. Jarrod Millman, Stéfan van der Walt, Ralf Gommers, Pauli „pv" Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke und Travis E. Oliphant. "Array programming with NumPy". *Nature* 585:357–362, 2020. London, England, UK: Springer Nature Limited. ISSN: 0028-0836. doi:10.1038/S41586-020-2649-2 (siehe S. 71–80, 125).

[24] Ian Hickson, Robin Berjon, Steve Faulkner, Travis Leithead, Erika Doyle Navara, Theresa O'Connor und Silvia Pfeiffer, Hrsg. *HTML5: A Vocabulary and Associated APIs for HTML and XHTML*. W3C Recommendation. Wakefield, MA, USA: World Wide Web Consortium (W3C), 28. Okt. 2014. URL: http://www.w3.org/TR/2014/REC-html5-20141028 (besucht am 2024-12-17) (siehe S. 124).

[25] Trey Hunner. *Python Morsels*. Reykjavík, Iceland: Python Morsels, 2025. URL: https://www.pythonmorsels.com (besucht am 2025-04-17) (siehe S. 99).

[26] John Hunt. *A Beginners Guide to Python 3 Programming*. 2. Aufl. Undergraduate Topics in Computer Science (UTICS). Cham, Switzerland: Springer, 2023. ISBN: 978-3-031-35121-1. doi:10.1007/978-3-031-35122-8 (siehe S. 94, 95, 125).

[27] John D. Hunter. "Matplotlib: A 2D Graphics Environment". *Computing in Science & Engineering* 9(3):90–95, Mai–Juni 2007. Piscataway, NJ, USA: Institute of Electrical and Electronics Engineers (IEEE). ISSN: 1521-9615. doi:10.1109/MCSE.2007.55 (siehe S. 71–80, 125).

References IV

[28] John D. Hunter, Darren Dale, Eric Firing, Michael Droettboom und The Matplotlib Development Team. *Matplotlib: Visualization with Python*. Austin, TX, USA: NumFOCUS, Inc., 2012–2025. URL: https://matplotlib.org (besucht am 2025-02-02) (siehe S. 125).

[29] Mike James. *Programmer's Python: Everything is an Object – Something Completely Different*. 2. Aufl. I/O Press, 25. Juni 2022 (siehe S. 94, 95).

[30] Robert Johansson. *Numerical Python: Scientific Computing and Data Science Applications with NumPy, SciPy and Matplotlib*. New York, NY, USA: Apress Media, LLC, Dez. 2018. ISBN: 978-1-4842-4246-9 (siehe S. 71–80, 97, 125).

[31] Stephen Curtis Johnson. *Lint, a C Program Checker*. Computing Science Technical Report 78–1273. New York, NY, USA: Bell Telephone Laboratories, Incorporated, 25. Okt. 1978. URL: https://wolfram.schneider.org/bsd/7thEdManVol2/lint/lint.pdf (besucht am 2024-08-23) (siehe S. 125).

[32] Bernd Klein. *Einführung in Python 3 – Für Ein- und Umsteiger*. 3., überarbeitete. München, Bayern, Germany: Carl Hanser Verlag GmbH & Co. KG, 2018. ISBN: 978-3-446-45208-4. doi:10.3139/9783446453876 (siehe S. 94, 95).

[33] Holger Krekel und pytest-Dev Team. *pytest Documentation*. Release 8.4. Freiburg, Baden-Württemberg, Germany: merlinux GmbH. URL: https://readthedocs.org/projects/pytest/downloads/pdf/latest (besucht am 2024-11-07) (siehe S. 125).

[34] Charles Landau. *TensorFlow Deep Dive: Build, Train, and Deploy Machine Learning Models with TensorFlow*. Sebastopol, CA, USA: O'Reilly Media, Inc., Dez. 2023 (siehe S. 71–80, 126).

[35] Kent D. Lee und Steve Hubbard. *Data Structures and Algorithms with Python*. Undergraduate Topics in Computer Science (UTICS). Cham, Switzerland: Springer, 2015. ISBN: 978-3-319-13071-2. doi:10.1007/978-3-319-13072-9 (siehe S. 125).

[36] Moritz Lenz. *Python Continuous Integration and Delivery: A Concise Guide with Examples*. New York, NY, USA: Apress Media, LLC, Dez. 2018. ISBN: 978-1-4842-4281-0 (siehe S. 96, 124).

[37] Reuven M. Lerner. *Pandas Workout*. Shelter Island, NY, USA: Manning Publications, Juni 2024. ISBN: 978-1-61729-972-8 (siehe S. 71–80, 125).

[38] Mark Lutz. *Learning Python*. 6. Aufl. Sebastopol, CA, USA: O'Reilly Media, Inc., März 2025. ISBN: 978-1-0981-7130-8 (siehe S. 94, 95, 125).

References V

[39] *Making Open Source Work Better for Developers: Highlights of the 2019 Tidelift Managed Open Source Survey*. Boston, MA, USA: Tidelift, Inc., Juni 2019. URL: https://tidelift.com/subscription/managed-open-source-survey (besucht am 2024-12-29) (siehe S. 61–69).

[40] Charlie Marsh. "Ruff". In: URL: https://pypi.org/project/ruff (besucht am 2025-08-29) (siehe S. 125).

[41] Charlie Marsh. *ruff: An Extremely Fast Python Linter and Code Formatter, Written in Rust*. New York, NY, USA: Astral Software Inc., 28. Aug. 2022. URL: https://docs.astral.sh/ruff (besucht am 2024-08-23) (siehe S. 125).

[42] Eric Matthes. *Python Crash Course*. 3. Aufl. San Francisco, CA, USA: No Starch Press, Jan. 2023. ISBN: 978-1-7185-0270-3 (siehe S. 94, 95).

[43] Aaron Maxwell. *What are f-strings in Python and how can I use them?* Oakville, ON, Canada: Infinite Skills Inc, Juni 2017. ISBN: 978-1-4919-9486-3 (siehe S. 98, 124).

[44] Wes McKinney. *Python for Data Analysis*. 3. Aufl. Sebastopol, CA, USA: O'Reilly Media, Inc., Aug. 2022. ISBN: 978-1-0981-0403-0 (siehe S. 97).

[45] Pedro Mejia Alvarez, Raul E. Gonzalez Torres und Susana Ortega Cisneros. *Exception Handling – Fundamentals and Programming*. SpringerBriefs in Computer Science. Cham, Switzerland: Springer, Feb. 2024. ISSN: 2191-5768. ISBN: 978-3-031-50680-2. doi:10.1007/978-3-031-50681-9 (siehe S. 61–69).

[46] NumPy Team. *NumPy*. San Francisco, CA, USA: GitHub Inc und Austin, TX, USA: NumFOCUS, Inc. URL: https://numpy.org (besucht am 2025-02-02) (siehe S. 125).

[47] A. Jefferson Offutt. "Unit Testing Versus Integration Testing". In: *Test: Faster, Better, Sooner – IEEE International Test Conference (ITC'1991)*. 26.–30. Okt. 1991, Nashville, TN, USA. Los Alamitos, CA, USA: IEEE Computer Society, 1991. Kap. Paper P2.3, S. 1108–1109. ISSN: 1089-3539. ISBN: 978-0-8186-9156-0. doi:10.1109/TEST.1991.519784 (siehe S. 126).

[48] Brian Okken. *Python Testing with pytest*. Flower Mound, TX, USA: Pragmatic Bookshelf by The Pragmatic Programmers, L.L.C., Feb. 2022. ISBN: 978-1-68050-860-4 (siehe S. 96, 125).

References VI

[49] Michael Olan. "Unit Testing: Test Early, Test Often". *Journal of Computing Sciences in Colleges (JCSC)* 19(2):319–328, Dez. 2003. New York, NY, USA: Association for Computing Machinery (ACM). ISSN: 1937-4771. doi:10.5555/948785.948830. URL: https://www.researchgate.net/publication/255673967 (besucht am 2025-09-05) (siehe S. 126).

[50] Ashwin Pajankar. *Hands-on Matplotlib: Learn Plotting and Visualizations with Python 3*. New York, NY, USA: Apress Media, LLC, Nov. 2021. ISBN: 978-1-4842-7410-1 (siehe S. 71–80, 97, 125).

[51] Ashwin Pajankar. *Python Unit Test Automation: Automate, Organize, and Execute Unit Tests in Python*. New York, NY, USA: Apress Media, LLC, Dez. 2021. ISBN: 978-1-4842-7854-3 (siehe S. 96, 125, 126).

[52] Pandas Developers. *Pandas*. Austin, TX, USA: NumFOCUS, Inc. und Montreal, QC, Canada: OVHcloud. URL: https://pandas.pydata.org (besucht am 2025-02-02) (siehe S. 125).

[53] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Z. Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai und Soumith Chintala. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems (NeurIPS'2019)*. 8.–14. Dez. 2019, Vancouver, BC, Canada. Hrsg. von Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox und Roman Garnett. San Diego, CA, USA: The Neural Information Processing Systems Foundation (NeurIPS), 2019, S. 8024–8035. ISBN: 978-1-7138-0793-3. URL: https://proceedings.neurips.cc/paper/2019/hash/bdbca288fee7f92f2bfa9f7012727740-Abstract.html (besucht am 2024-07-18) (siehe S. 71–80, 125).

[54] Fabian Pedregos, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake VanderPlas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot und Edouard Duchesnay. "Scikit-learn: Machine Learning in Python". *Journal of Machine Learning Research (JMLR)* 12:2825–2830, Okt. 2011. Cambridge, MA, USA: MIT Press. ISSN: 1532-4435. doi:10.5555/1953048.2078195 (siehe S. 71–80, 125).

References VII

[55] Yasset Pérez-Riverol, Laurent Gatto, Rui Wang, Timo Sachsenberg, Julian Uszkoreit, Felipe da Veiga Leprevost, Christian Fufezan, Tobias Ternent, Stephen J. Eglen, Daniel S. Katz, Tom J. Pollard, Alexander Konovalov, Robert M. Flight, Kai Blin und Juan Antonio Vizcaíno. "Ten Simple Rules for Taking Advantage of Git and GitHub". *PLOS Computational Biology* 12(7), 14. Juli 2016. San Francisco, CA, USA: Public Library of Science (PLOS). ISSN: 1553-7358. doi:10.1371/JOURNAL.PCBI.1004947 (siehe S. 124).

[56] *Programming Languages – C, Working Document of SC22/WG14*. International Standard ISO/IEC9899:2017 C17 Ballot N2176. Geneva, Switzerland: International Organization for Standardization (ISO) und International Electrotechnical Commission (IEC), Nov. 2017. URL: https://files.lhmouse.com/standards/ISO%20C%20N2176.pdf (besucht am 2024-06-29) (siehe S. 124).

[57] "programming: Meaning of *programming* in English". In: *Cambridge Dictionary English (UK)*. Cambridge, England, UK: Cambridge University Press & Assessment, Juni 2024. URL: https://dictionary.cambridge.org/dictionary/english/programming (besucht am 2024-06-17) (siehe S. 20, 21).

[58] *Python 3 Documentation*. Beaverton, OR, USA: Python Software Foundation (PSF), 2001–2025. URL: https://docs.python.org/3 (besucht am 2024-07-05) (siehe S. 99).

[59] Sebastian Raschka, Yuxi Liu und Vahid Mirjalili. *Machine Learning with PyTorch and Scikit-learn*. Birmingham, England, UK: Packt Publishing Ltd, Feb. 2022. ISBN: 978-1-80181-931-2 (siehe S. 71–80, 125).

[60] Abhishek Ratan, Eric Chou, Pradeeban Kathiravelu und Dr. M.O. Faruque Sarker. *Python Network Programming*. Birmingham, England, UK: Packt Publishing Ltd, Jan. 2019. ISBN: 978-1-78883-546-6 (siehe S. 98).

[61] *Real Python Tutorials*. Vancouver, BC, Canada: DevCademy Media Inc., 2021–2025. URL: https://realpython.com (besucht am 2025-04-17) (siehe S. 99).

[62] Kristian Rother. *Pro Python Best Practices: Debugging, Testing and Maintenance*. New York, NY, USA: Apress Media, LLC, März 2017. ISBN: 978-1-4842-2241-6 (siehe S. 96).

[63] Per Runeson. "A Survey of Unit Testing Practices". *IEEE Software* 23(4):22–29, Juli–Aug. 2006. Piscataway, NJ, USA: Institute of Electrical and Electronics Engineers (IEEE). ISSN: 0740-7459. doi:10.1109/MS.2006.91 (siehe S. 126).

References VIII

[64] Stuart J. Russell und Peter Norvig. *Artificial Intelligence: A Modern Approach (AIMA)*. 4. Aufl. Hoboken, NJ, USA: Pearson Education, Inc. ISBN: 978-1-292-40113-3. URL: https://aima.cs.berkeley.edu (besucht am 2024-06-27) (siehe S. 71–80, 124).

[65] Yeonhee Ryou, Sangwoo Joh, Joonmo Yang, Sujin Kim und Youil Kim. "Code Understanding Linter to Detect Variable Misuse". In: *37th IEEE/ACM International Conference on Automated Software Engineering (ASE'2022)*. 10.–14. Okt. 2022, Rochester, MI, USA. New York, NY, USA: Association for Computing Machinery (ACM), 2022, 133:1–133:5. ISBN: 978-1-4503-9475-8. doi:10.1145/3551349.3559497 (siehe S. 125).

[66] Shai Shalev-Shwartz und Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge, England, UK: Cambridge University Press (CUP), Juli 2014. ISBN: 978-1-107-05713-5. URL: http://www.cs.huji.ac.il/~shais/UnderstandingMachineLearning (besucht am 2024-06-27) (siehe S. 71–80, 125).

[67] Anna Skoulikari. *Learning Git*. Sebastopol, CA, USA: O'Reilly Media, Inc., Mai 2023. ISBN: 978-1-0981-3391-7 (siehe S. 124).

[68] Brett Slatkin. *Effective Python: 125 Specific Ways to Write Better Python*. 3. Aufl. Reading, MA, USA: Addison-Wesley Professional, Nov. 2024. ISBN: 978-0-13-817239-8 (siehe S. 94, 95).

[69] Eric V. „ericvsmith" Smith. *Literal String Interpolation*. Python Enhancement Proposal (PEP) 498. Beaverton, OR, USA: Python Software Foundation (PSF), 6. Nov. 2016–9. Sep. 2023. URL: https://peps.python.org/pep-0498 (besucht am 2024-07-25) (siehe S. 124).

[70] "Stack Overflow 2024 Developer Survey". In: *Stack Overflow*. New York, NY, USA: Stack Exchange Inc., Mai–Juni 2024. URL: https://survey.stackoverflow.co/2024 (besucht am 2025-06-01) (siehe S. 71–80).

[71] The PEP Editors. *Index of Python Enhancement Proposals (PEPs)*. Python Enhancement Proposal (PEP) 0. Beaverton, OR, USA: Python Software Foundation (PSF), 13. Juli 2000. URL: https://peps.python.org (besucht am 2025-04-17) (siehe S. 99).

[72] George K. Thiruvathukal, Konstantin Läufer und Benjamin Gonzalez. "Unit Testing Considered Useful". *Computing in Science & Engineering* 8(6):76–87, Nov.–Dez. 2006. Piscataway, NJ, USA: Institute of Electrical and Electronics Engineers (IEEE). ISSN: 1521-9615. doi:10.1109/MCSE.2006.124. URL: https://www.researchgate.net/publication/220094077 (besucht am 2024-10-01) (siehe S. 126).

[73] Brad Traversy. *Modern HTML & CSS From The Beginning 2.0*. 2. Aufl. Birmingham, England, UK: Packt Publishing Ltd, Juli 2024. ISBN: 978-1-83588-056-2 (siehe S. 124).

References IX

[74] Mariot Tsitoara. *Beginning Git and GitHub: Version Control, Project Management and Teamwork for the New Developer*. New York, NY, USA: Apress Media, LLC, März 2024. ISBN: 979-8-8688-0215-7 (siehe S. 124, 126).

[75] Guido van Rossum. *Computer Programming for Everybody (Revised Proposal). A Scouting Expedition for the Programmers of Tomorrow*. CNRI Proposal 90120-1a. Reston, VA, USA: Corporation for National Research Initiatives (CNRI), Juli 1999. URL: https://www.python.org/doc/essays/cp4e (besucht am 2024-06-27) (siehe S. 71–80).

[76] Guido van Rossum, Barry Warsaw und Alyssa Coghlan. *Style Guide for Python Code*. Python Enhancement Proposal (PEP) 8. Beaverton, OR, USA: Python Software Foundation (PSF), 5. Juli 2001. URL: https://peps.python.org/pep-0008 (besucht am 2024-07-27) (siehe S. 33–49).

[77] Pauli „pv“ Virtanen, Ralf Gommers, Travis E. Oliphant, Matt „mdhaber“ Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, CJ Carey, Ilhan „ilayn“ Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregos, Paul van Mulbregt und SciPy 1.0 Contributors. “SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python”. *Nature Methods* 17:261–272, 2. März 2020. London, England, UK: Springer Nature Limited. ISSN: 1548-7091. doi:10.1038/s41592-019-0686-2. URL: http://arxiv.org/abs/1907.10121 (besucht am 2024-06-26). See also arXiv:1907.10121v1 [cs.MS] 23 Jul 2019. (Siehe S. 71–80, 125).

[78] *W3Schools: Python Tutorials*. Sandnes, Rogaland, Norway: Refsnes Data AS, 1999–2025. URL: https://www.w3schools.com/python (besucht am 2025-04-17) (siehe S. 99).

[79] Thomas Weise (汤卫思). *Programming with Python*. Hefei, Anhui, China (中国安徽省合肥市): Hefei University (合肥大学), School of Artificial Intelligence and Big Data (人工智能与大数据学院), Institute of Applied Optimization (应用优化研究所, IAO), 2024–2025. URL: https://thomasweise.github.io/programmingWithPython (besucht am 2025-01-05) (siehe S. 94–99, 125).

[80] Thomas Weise (汤卫思) und Zhize Wu (吴志泽). “Replicable Self-Documenting Experiments with Arbitrary Search Spaces and Algorithms”. In: *Conference on Genetic and Evolutionary Computation (GECCO'2023), Companion Volume*. 15.–19. Juli 2023, Lisbon, Portugal. Hrsg. von Sara Silva und Luís Paquete. New York, NY, USA: Association for Computing Machinery (ACM), 2023, S. 1891–1899. ISBN: 979-8-4007-0120-7. doi:10.1145/3583133.3596306 (siehe S. 71–80, 125).

References X

[81] Kevin Wilson. *Python Made Easy*. Birmingham, England, UK: Packt Publishing Ltd, Aug. 2024. ISBN: 978-1-83664-615-0 (siehe S. 94, 95, 125).

[82] Dmitry Zinoviev. *Discrete Event Simulation: It's Easy with SimPy!* arXiv.org: Computing Research Repository (CoRR) abs/2405.01562. Ithaca, NY, USA: Cornell Universiy Library, 3. Apr. 2024. doi:10.48550/ARXIV.2405.01562. URL: https://arxiv.org/abs/2405.01562 (besucht am 2024-06-27). arXiv:2405.01562v1 [cs.MS] 3 Apr 2024 (siehe S. 71–80, 125).

Glossary (in English) I

AI Artificial Intelligence, see, e.g.,[64]

API An *Application Programming Interface* is a set of rules or protocols that enables one software application or component to use or communicate with another[19].

C is a programming language, which is very successful in system programming situations[14,56].

CI *Continuous Integration* is a software development process where developers integrate new code into a codebase hosted in a Version Control Systems (VCS), after which automated tools run an automated build process including code analysis (such as linters) and unit test execution[36]. If the build succeeds and no errors or problems with the code are, the code may automatically be deployed (if the CI system is configured to do so).

DS Data Science, see, e.g.,[22].

f-string let you include the results of expressions in strings[7,16,17,20,43,69]. They can contain expressions (in curly braces) like `f"a{6-1}b"` that are then transformed to text via (string) interpolation, which turns the string to `"a5b"`. F-strings are delimited by `f"..."`.

Git is a distributed Version Control Systems (VCS) which allows multiple users to work on the same code while preserving the history of the code changes[67,74]. Learn more at `https://git-scm.com`.

GitHub is a website where software projects can be hosted and managed via the Git VCS[55,74]. Learn more at `https://github.com`.

HTML The Hyper Text Markup Language (HTML) is the text format used by the World Wide Web (WWW)[5,24,73].

JavaScript JavaScript is the predominant programming language used in websites to develop interactive contents for display in browsers[15].

Glossary (in English) II

linter A linter is a tool for analyzing program code to identify bugs, problems, vulnerabilities, and inconsistent code styles[31,65]. Ruff is an example for a linter used in the Python world.

Matplotlib is a Python package for plotting diagrams and charts[27,28,30,50]. Learn more at at https://matplotlib.org[28].

ML Machine Learning, see, e.g.,[66]

moptipy is the *Metaheuristic Optimization in Python* library[80]. Learn more at https://thomasweise.github.io/moptipy.

NumPy is a fundamental package for scientific computing with Python, which offers efficient array datastructures[10,23,30]. Learn more at https://numpy.org[46].

Pandas is a Python data analysis and manipulation library[3,37]. Learn more at https://pandas.pydata.org[52].

pytest is a framework for writing and executing unit tests in Python[12,33,48,51,81]. Learn more at https://pytest.org.

Python The Python programming language[26,35,38,79], i.e., what you will learn about in our book[79]. Learn more at https://python.org.

PyTorch is a Python library for deep learning and AI[53,59]. Learn more at https://pytorch.org.

Ruff is a linter and code formatting tool for Python[40,41]. Learn more at https://docs.astral.sh/ruff or in[79].

Scikit-learn is a Python library offering various machine learning tools[54,59]. Learn more at https://scikit-learn.org.

SciPy is a Python library for scientific computing[30,77]. Learn more at https://scipy.org.

SimPy is a Python library for discrete event simulation[82]. Learn more at https://simpy.readthedocs.io.

Glossary (in English) III

(string) interpolation In Python, string interpolation is the process where all the expressions in an f-string are evaluated and the final string is constructed. An example for string interpolation is turning `f"Rounded {1.234:.2f}"` to `"Rounded 1.23"`.

TensorFlow is a Python library for implementing machine learning, especially suitable for training of neural networks[1,34]. Learn more at `https://www.tensorflow.org`.

unit test Software development is centered around creating the program code of an application, library, or otherwise useful system. A *unit test* is an *additional* code fragment that is not part of that productive code. It exists to execute (a part of) the productive code in a certain scenario (e.g., with specific parameters), to observe the behavior of that code, and to compare whether this behavior meets the specification[4,47,49,51,63,72]. If not, the unit test fails. The use of unit tests is at least threefold: First, they help us to detect errors in the code. Second, program code is usually not developed only once and, from then on, used without change indefinitely. Instead, programs are often updated, improved, extended, and maintained over a long time. Unit tests can help us to detect whether such changes in the program code, maybe after years, violate the specification or, maybe, cause another, depending, module of the program to violate its specification. Third, they are part of the documentation or even specification of a program.

VCS A *Version Control System* is a software which allows you to manage and preserve the historical development of your program code[74]. A distributed VCS allows multiple users to work on the same code and upload their changes to the server, which then preserves the change history. The most popular distributed VCS is Git.

WWW World Wide Web[5,9]