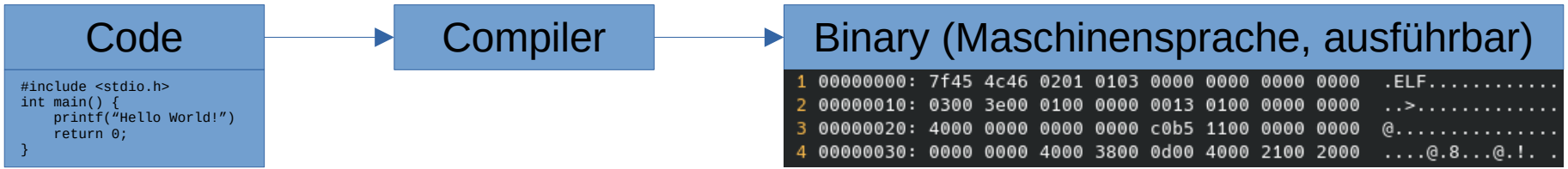
Einführung in Python

Jonas Pleyer
Jonas.pleyer@fdm.uni-freiburg.de
www.fdm.uni-freiburg.de/Members/spatsysbio/Members/JonasPleyer

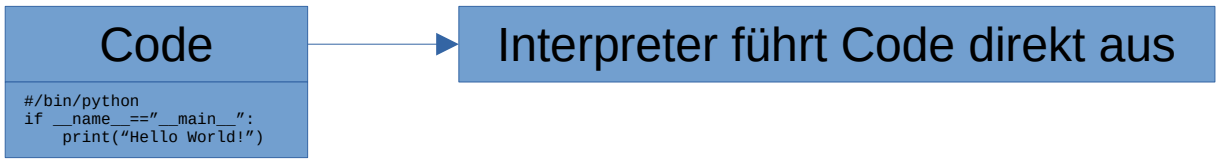
Unterschiede von Sprachen

Kompillierte Sprachen



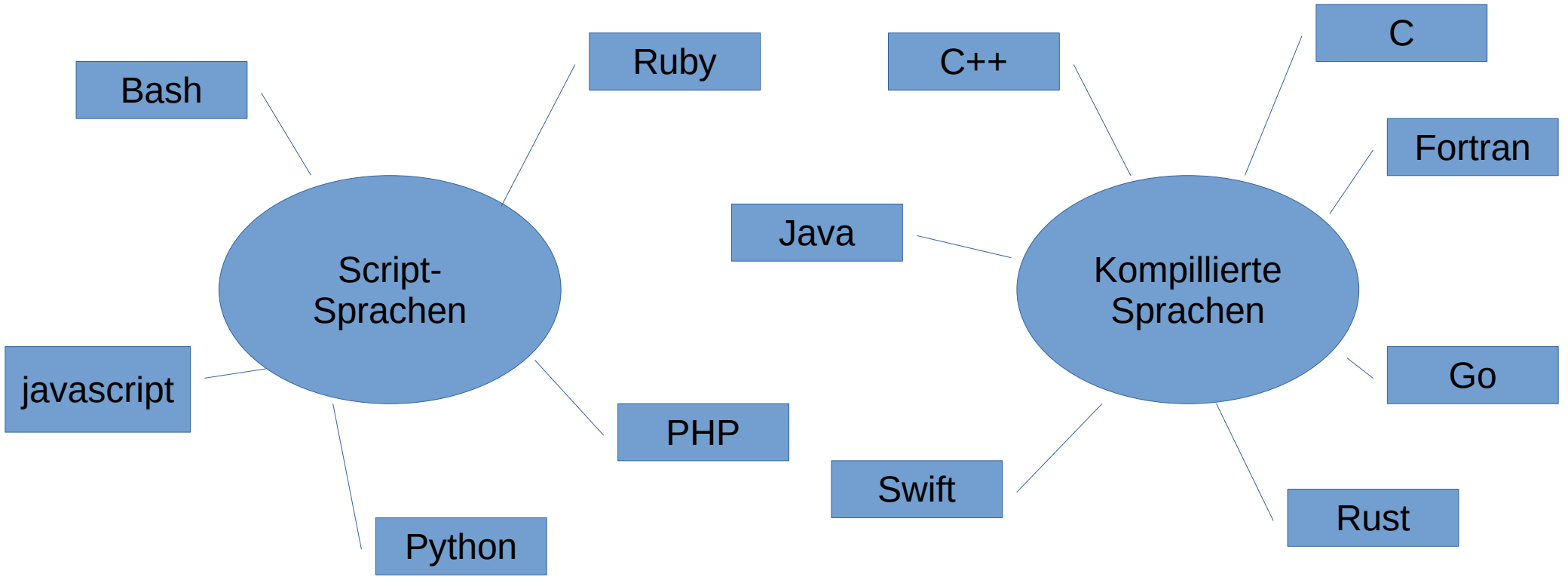
- Code-Text → Compiler → binary (kann auf PC ausgeführt werden)
- Meist bessere Performance als Script-Sprachen (~10-100x)
- Fehler im Code werden zum Teil vor Ausführung bemerkt

Script-Sprachen

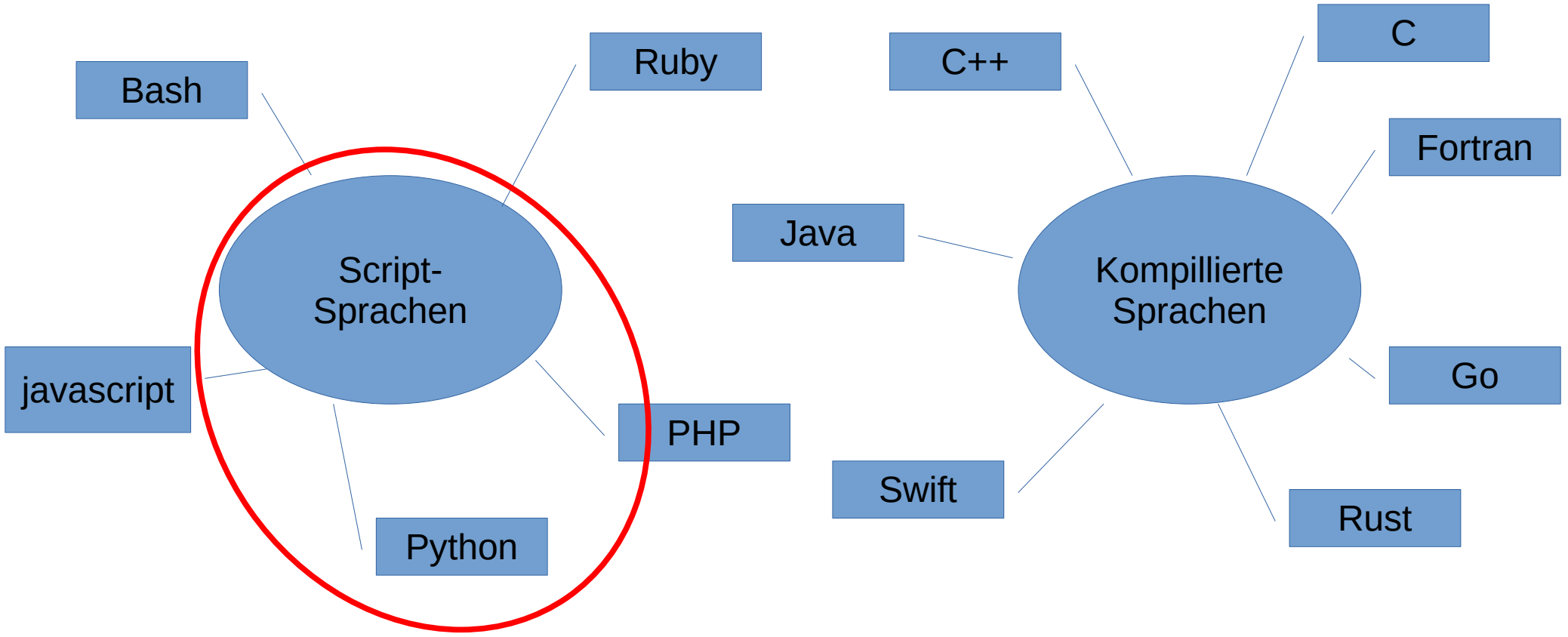


- Code-Text → Interpreter führt Code direkt aus
- Schnellere Entwicklung von Code
- Können teilweise bestehenden Code von kompillierten Sprachen verwenden
- Fehler im Code werden erst beim Ausführen bemerkt

Einige Programmiersprachen



Einige Programmiersprachen



Python

- High-level Script-Sprache
- Designed 20.02.1991 – Guido van Rossum (“benevolent dictator for life”)
  Seit 2019 Python “steering council”
- Web-Development, Data-Science, Scientific Computing, AI, ...
- Libraries für spezielle Anwendungen
  Standard library → allgemeine Datenstrukturen und Algorithmen
  Numpy → numerische Anwendungen
  Scipy → wissenschaftliche Algorithmen etc.
  Matplotlib → erzeugen von Grafiken
  ...
- Verschiedene Versionen sind nicht kompatibel!
  - Python 2.XX
  - Python 3.XX
- Verschiedene Programmierumgebungen (Development Environment)

Python

Programmierumgebungen

- Texteditor (mit Syntax-Highlighting)
- VSCode + Command Line (persöhnliche Wahl)
- Spyder
- PyCharm
- Jupyter-Notebooks (andere Form)
- …

Wählt eine Möglichkeit, solange es funktioniert. Nach heute sollte es im besten falle keine Probleme mehr geben.

Paketinstallation

- Pip (Standard)
- Conda (advanced)
- Paketmanager (Linux)

Python – Bsp: Hello World

```
#!/usr/bin/env python3

if __name__ == “__main__”:
     print(“Hello World!”)
```

Output

```
Hello World!
```

Python – Bsp: Variablen

```
#!/usr/bin/env python3

if __name__ == “__main__”:
     # Erstelle eine Variable mit ganzzahligem Wert 1
     n = 1
     print(n)

     # Eine Variable mit dem Wert “Wort”
     w = “Wort”
     print(w)

     # Variable mit Fließkommazahl
     f = 3.1415
     print(f)

     # Erstelle ein Tupel aus mehreren Variablen
     t = (n, w, f)
     print(t)
```

Output

```
1

Wort

3.1415

(1, ‘Wort’, 3.1415)
```

Python – Bsp: Zahlen

```
#!/usr/bin/env python3

if __name__ == "__main__":
    # Erstelle eine Variable mit ganzzahligem Wert 10
    n = 10
    print(n)

    # Einfache Mathematik
    a = n*2 + 3 - 400
    print(a)

    # Exponenten berechnen n^3
    b = n**3
    print(b)

    # Fließkommazahlen
    c = 3.1415/99
    print(c)

    # Selbstzuweisung
    c += 2
    print(c)
```

Output

```
10

-377

1000

0.03173232323232324

2.0317323232323234
```

Python – Bsp: Listen 1

```
#!/usr/bin/env python3

if __name__ == “__main__”:
     # Erstelle eine Liste mit 7 Elementen
     a = [0, 1, 2, 3, 4, 5, 6]

     # Länge der Liste
     print(len(a))

     # ?-te Element der Liste
     print(a[1])

     # Alle Elemente von 0 bis 3 exclusive Element 4!
     print(a[0:3])

     # Das letzte Element
     print(a[-1])

     # Das vorletzte Element
     print(a[-2])
```

Output

```
7

1

[0, 1, 2]

6

5
```

Python – Bsp: Listen 2

```
#!/usr/bin/env python3

if __name__ == “__main__”:
     # Erstelle zwei Liste mit 7 Elementen
     a = [0, 1, 2, 3]
     b = ['null', 'eins', 'zwei', 'drei’]

     # Verkette beide Listen
     c = a + b
     # Gebe das Ergebnis aus
     print(c)

     # Verdoppele beide Listen
     a = 2*a
     b = 2*b
     # Gebe das Ergebnis aus
     print(a)

     print(b)
```

Output

```
[0, 1, 2, 3, 'null', 'eins', 'zwei', 'drei']

[0, 1, 2, 3, 0, 1, 2, 3]

['null', 'eins', 'zwei', 'drei', 'null',
'eins', 'zwei', 'drei']
```

Python – Bsp: Listen 3

```
#!/usr/bin/env python3

if __name__ == “__main__”:
     # Erstelle eine Liste mit 7 Elementen
     a = [0, 1, 2, 3]

     # neue Liste mit veränderten Werten
     b = [elem*2 for elem in a]
     print(b)

     m = max(b)
     print(m)
```

Output

```
[0, 2, 4, 6]

6
```

Python – Bsp: Strings

```
#!/usr/bin/env python3

if __name__ == "__main__":
    # Eine String-Variable
    w = "Das hier ist ein ganzer Satz."

    # Erster Buchstabe
    print(w[0])

    # Letzter Buchstabe
    print(w[-1])

    # Teile an bestimmtem Zeichen auf
    print(w.split(" "))

    # Tausche bestimmte Zeichen
    print(w.replace("e", "aiai"))
```

Output

```
D

.

['Das', 'hier', 'ist', 'ein', 'ganzer',
'Satz.']

Das hiaiair ist aiaiin ganzaiair Satz.
```

Python – Bsp: Schleifen

```
#!/usr/bin/env python3

if __name__ == "__main__":
    # For-schleifen
    # Schleife über eine Liste
    a = ["das", "ist", 1, "Test"]
    for element in a:
        print(element)

    # Schleife über eine range von Zahlen
    for i in range(4):
        print(i)

    # Tricks
    # enumerate, um den index der liste zu bekommen
    for i, element in enumerate(a):
        print("Index:", i, "Inhalt:", element)
```

Output

```
das
ist
1
Test

0
1
2
3

Index: 0 Inhalt: das
Index: 1 Inhalt: ist
Index: 2 Inhalt: 1
Index: 3 Inhalt: Test
```

Python – Bsp: Funktionen1

```
#!/usr/bin/env python3


# Funktion ohne parameter
def gibt_schoene_sachen_aus():
    print("Schöne Sachen")


# Funktion mit input parametern
def sag_mir_quadrat(zahl):
    print(zahl**2)


if __name__ == "__main__":
    # Rufe eine funktion ohne parameter auf
    gibt_schoene_sachen_aus()

    # Rufe funktion mit input parametern auf
    sag_mir_quadrat(3)
```

Output

```
Schöne Sachen

9
```

Python – Bsp: Funktionen2

```
#!/usr/bin/env python3


# Funktion mit output
def gib_mir_zahl():
    return 2


# berechnet aus Dichte und Volumen die Masse
def masse(dichte, volumen):
    return dichte*volumen


if __name__ == "__main__":
    # Funktion mit output
    zahl = gib_mir_zahl()

    # Berechne die masse von 1000l wasser mit dichte 1kg/l
    rho = 1.0
    V = 1000.0
    print(masse(rho, V))
```

Output

```
1000.0
```

Python – Bsp: Libraries

```
#!/usr/bin/env python3

# Importiere library für numerische operationen
import numpy as np
import matplotlib.pyplot as plt


if __name__ == "__main__":
    # Benutze die libraries
    # Dokumentation von funktionen -> website der library
    x1 = np.arange(0, 7)
    x2 = np.linspace(0,7)
    # Berechne den cosinus
    y1 = np.cos(x1)
    y2 = np.cos(x2)

    # Plotte das Ergebnis
    plt.plot(x1, y1)
    plt.plot(x2, y2)
    plt.show()
```

Output