

Kompilieren & Interpretieren

Arten von Code

Quellcode

- lesbarer Text eines Computerprogrammes oder einer Webseite
- wird in einer Programmiersprache verfasst
- kann aus mehreren Dateien bestehen
- Syntax und Befehle hängt von der Sprache ab
- vom Programmierer geschrieben
- über Editor/Entwicklungsumgebung erstellt

```
main.py x
1  import random
2
3
4  def main():
5      show_header()
6      play_game("You", "Computer")
7
8
9  def show_header():
10     print("-----")
11     print(" Rock Paper Scissors")
12     print("-----")
13
14
15  def play_game(player_1, player_2):...
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55  def check_for_winning_throw(player_1, player_2, roll1, roll2):...
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89  def get_roll(player_name, rolls):
90     print("Available rolls:")
91     for index, r in enumerate(rolls, start=1):
92         print(f"{index}. {r}")
93
94     text = input(f"{player_name}, what is your roll? ")
95     selected_index = int(text) - 1
96
97     if selected_index < 0 or selected_index >= len(rolls):
98         print(f"Sorry {player_name}, {text} is out of bounds!")
99         return None
100
101     return rolls[selected_index]
102
103
104  if __name__ == '__main__':
105     main()
```

Arten von Code

Maschinencode



Quelle: <http://evlearners.com/binary-arithmetic-tutorial/> [Stand: 08.09.2022]

- die „Programmiersprache eines Computers“
- Bits und Bytes
- Beispiel: 1011 1100 1101 0101 0000
- Stark abhängig von der Hardware
- für Menschen unverständlich
- wird vom Computer ausgeführt

Arten von Code

Bytecode



Quelle: <http://evlearners.com/binary-arithmetic-tutorial/> [Stand: 08.09.2022]

- Zwischencode
- Bits und Bytes
 - Beispiel: 1101 0001 1111 1110 0001
- Sammlung von Befehlen für eine virtuelle Maschine
- unabhängig von Hardware
- für Menschen unverständlich

Vom Quellcode zum Maschinencode

Quellcode

Vom Menschen lesbar



Quellcode wird übersetzt.

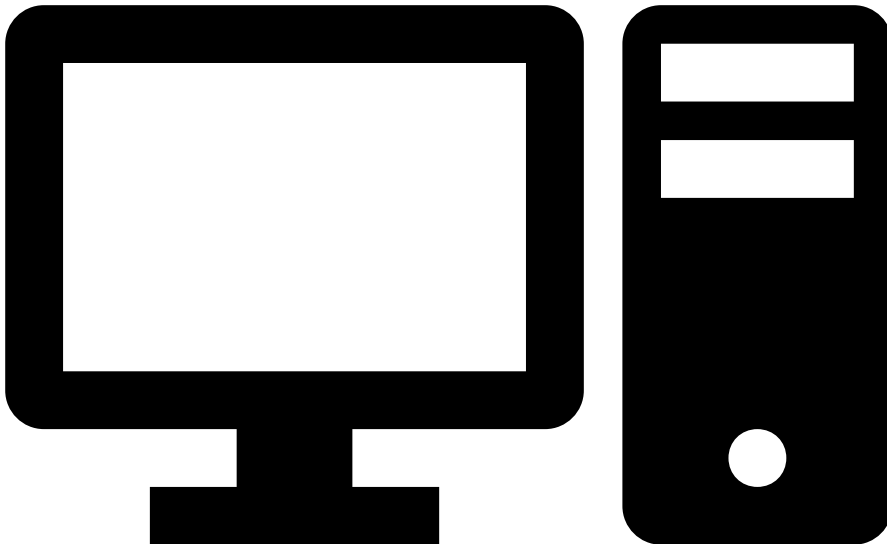
1. Compiler
2. Interpreter

Maschinencode

Vom Computer lesbar

Compiler

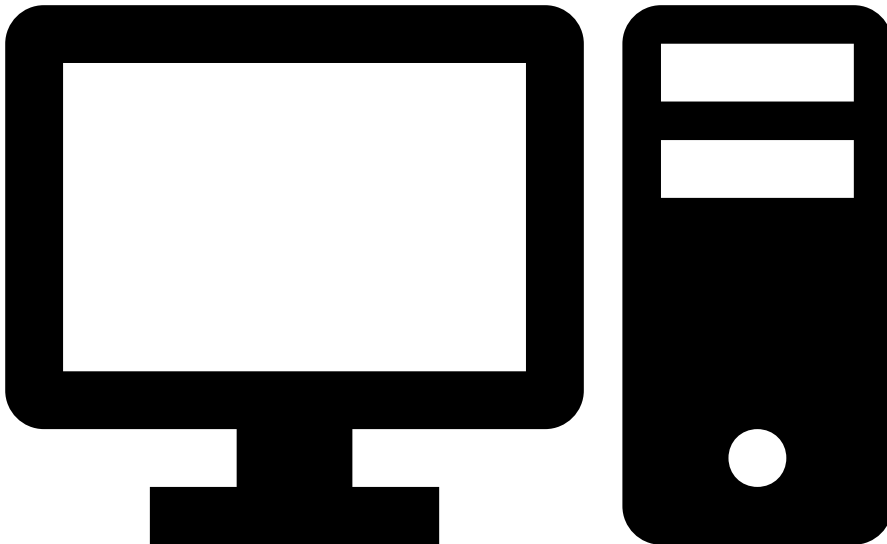
- Übersetzt kompletten Quellcode in Maschinencode
 - Genannt: Kompilieren
- Erst dann kann Code ausgeführt werden !



```
IF X >= 5 THEN  
  MSGBOX «Geben Sie eine Zahl <5 ein»  
ELSE  
  MSGBOX «Perfekt»  
END IF
```

Compiler

- Übersetzt kompletten Quellcode in Maschinencode
 - Genannt: Kompilieren
- Erst dann kann Code ausgeführt werden !



01101010

MSGBOX «Geben Sie eine Zahl <5 ein»

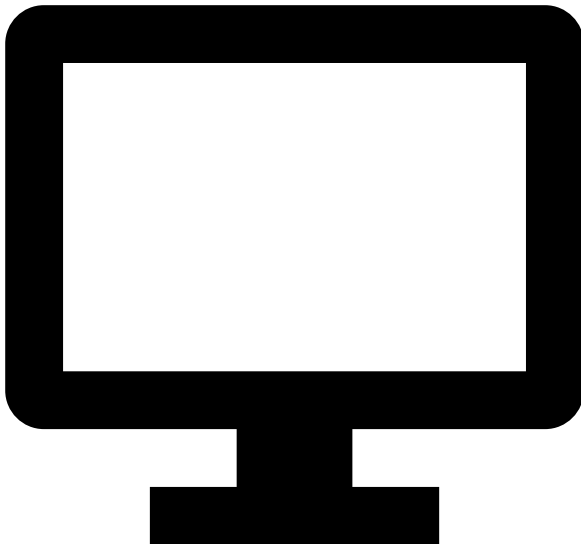
ELSE

MSGBOX «Perfekt»

END IF

Compiler

- Übersetzt kompletten Quellcode in Maschinencode
 - Genannt: Kompilieren
- Erst dann kann Code ausgeführt werden !



01101010

10001111

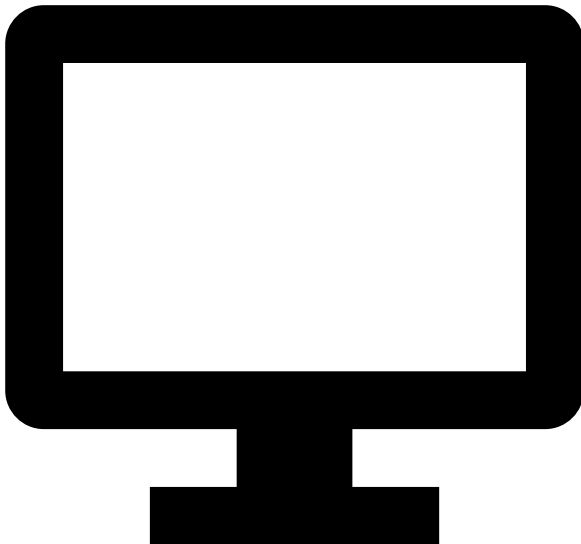
ELSE

MSGBOX «Perfekt»

END IF

Compiler

- Übersetzt kompletten Quellcode in Maschinencode
 - Genannt: Kompilieren
- Erst dann kann Code ausgeführt werden !



01101010

10001111

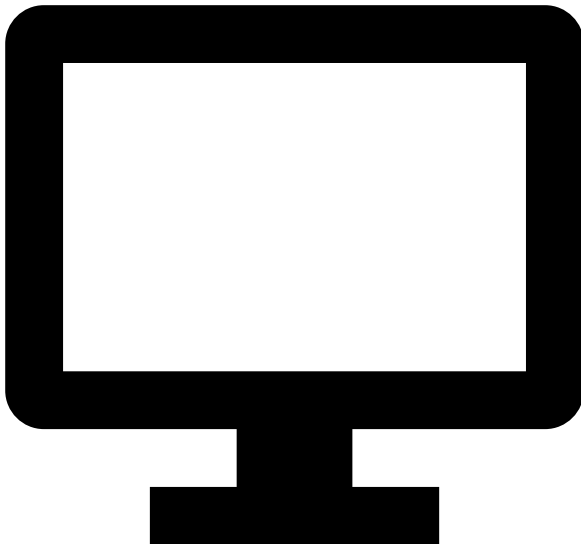
11000010

MSGBOX «Perfekt»

END IF

Compiler

- Übersetzt kompletten Quellcode in Maschinencode
 - Genannt: Kompilieren
- Erst dann kann Code ausgeführt werden !



01101010

10001111

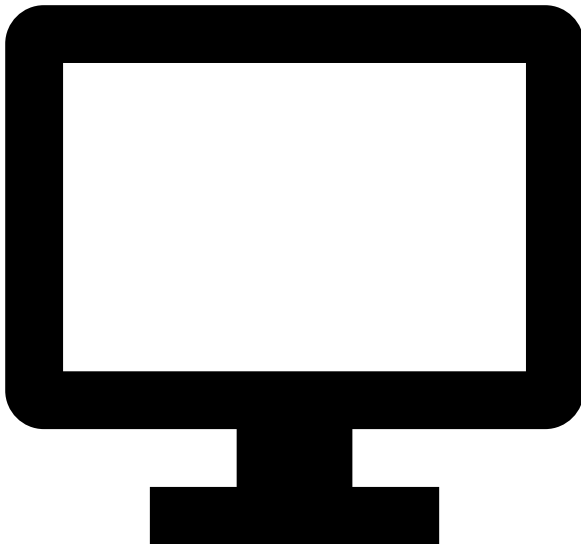
11000010

01000001

END IF

Compiler

- Übersetzt kompletten Quellcode in Maschinencode
 - Genannt: Kompilieren
- Erst dann kann Code ausgeführt werden !



01101010

10001111

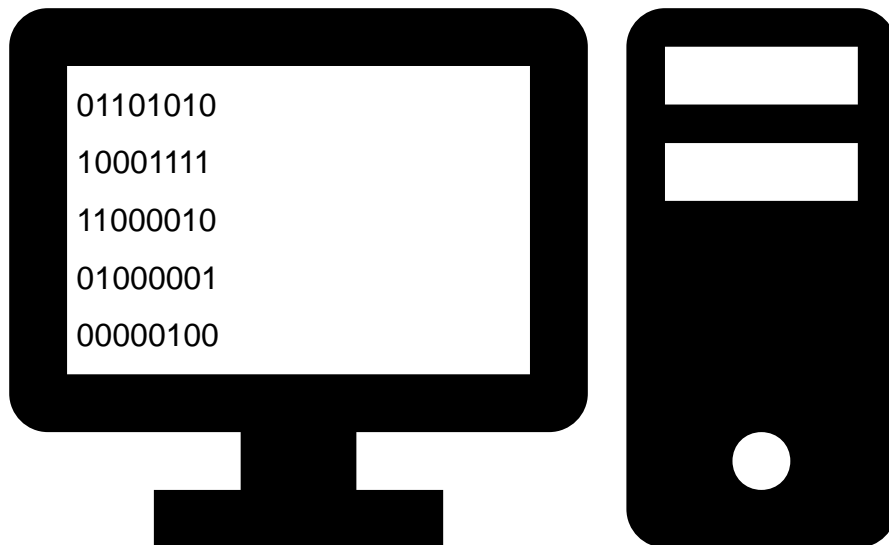
11000010

01000001

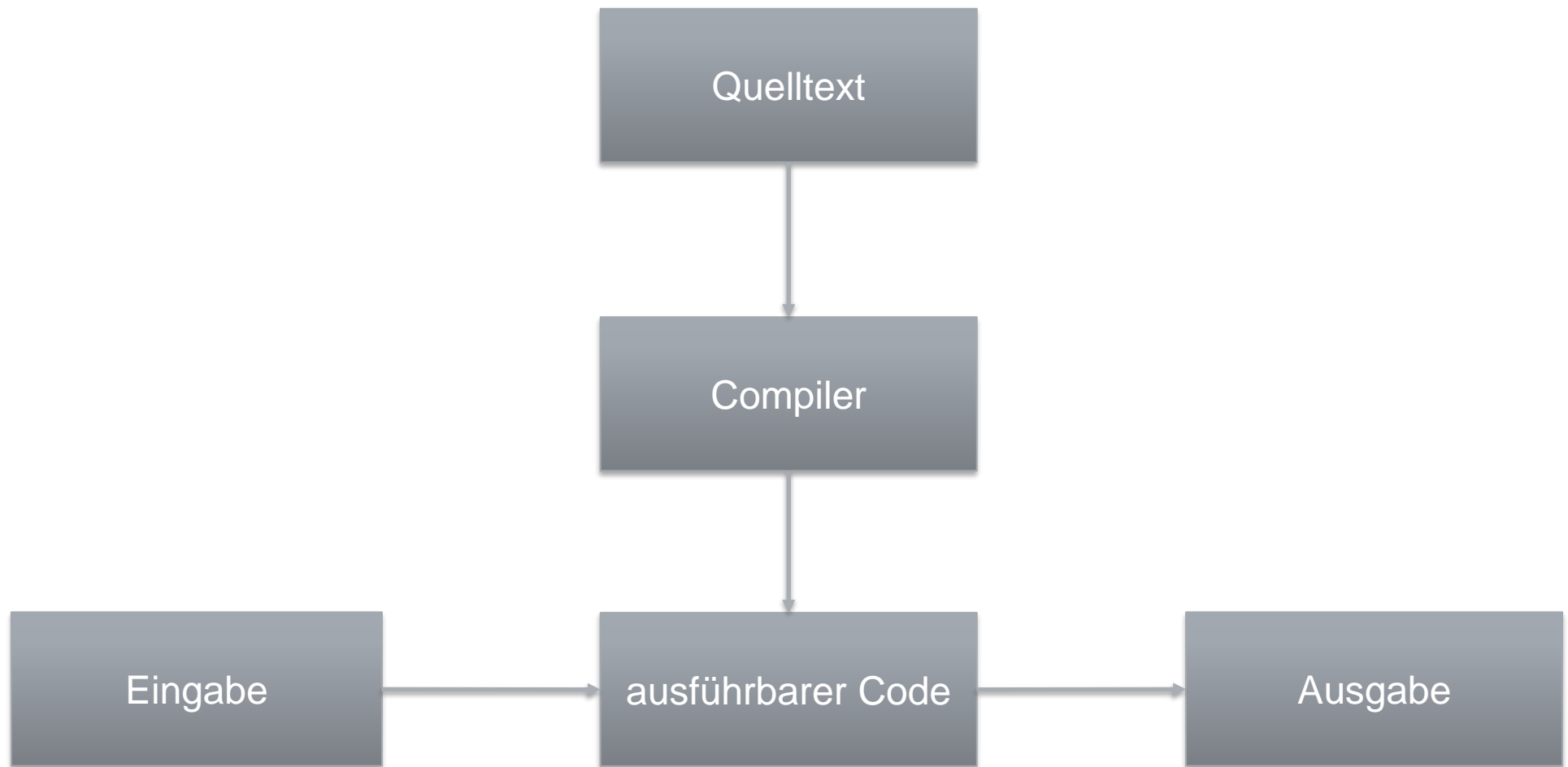
00000100

Compiler

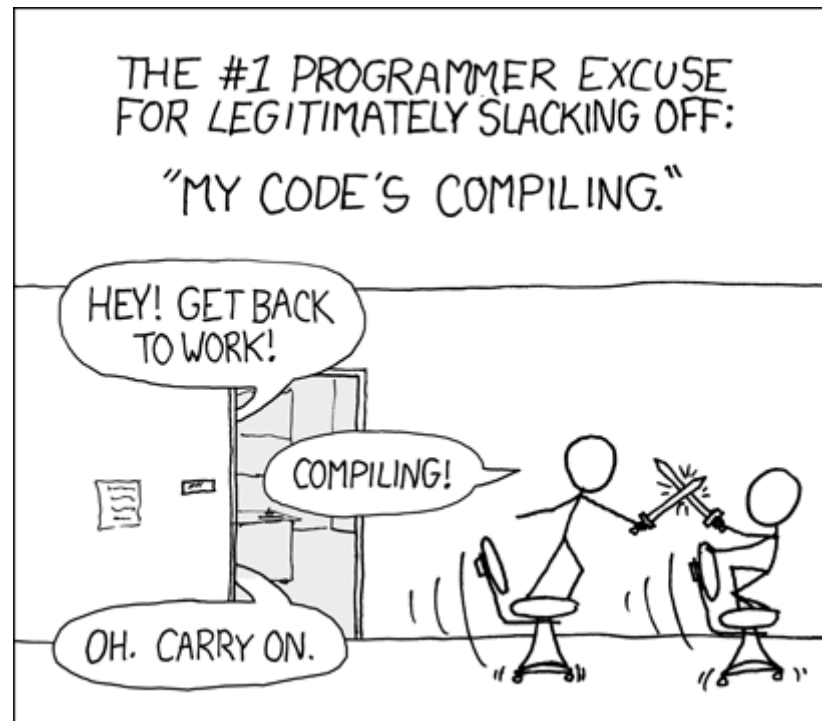
- Übersetzt kompletten Quellcode in Maschinencode
 - Genannt: Kompilieren
- Erst dann kann Code ausgeführt werden !



Compiler



Compiler

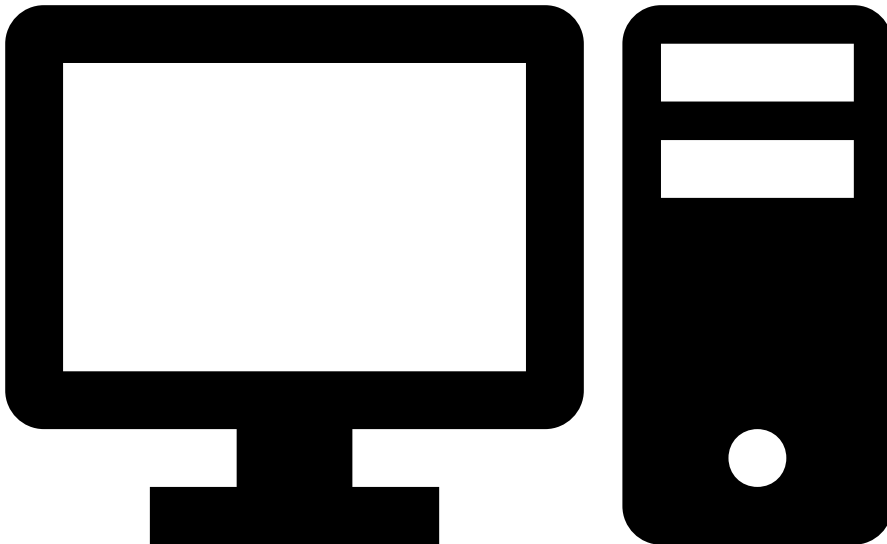


'Are you stealing those LCDs?' 'Yeah, but I'm doing it while my code compiles.'

Quelle: <https://xkcd.com/303/> [Stand: 15.02.2022]

Interpreter

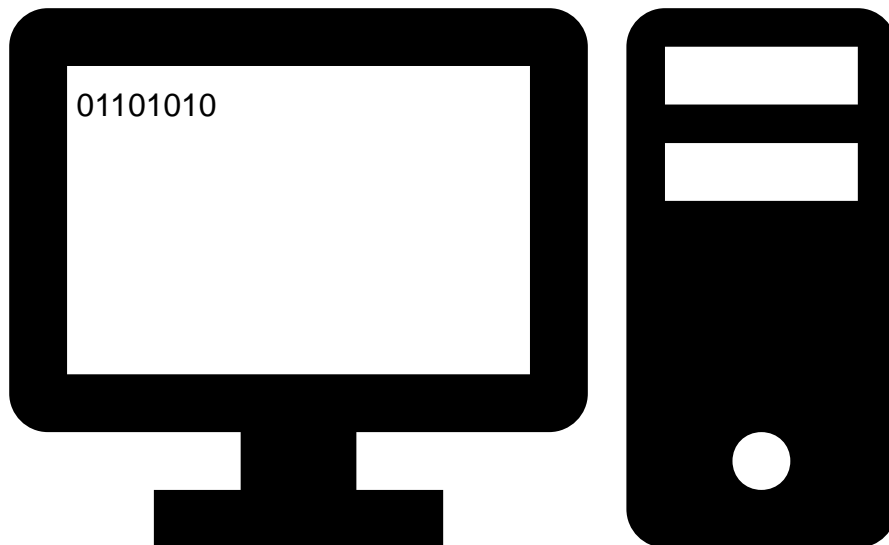
- Übersetzt einzelne Anweisungen in Maschinencode während Ausführung
- Durchführung läuft „zeilenweise“ ab



```
IF X > 5 THEN  
  MSGBOX «Geben Sie eine Zahl <5 ein»  
ELSE  
  MSGBOX «Perfekt»  
END IF
```

Interpreter

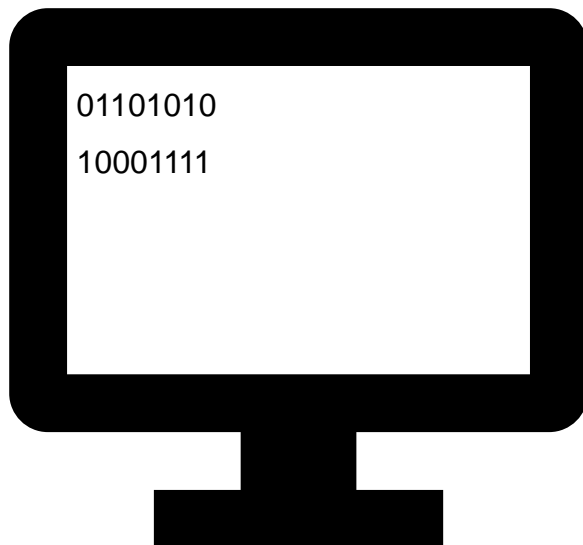
- Übersetzt einzelne Anweisungen in Maschinencode während Ausführung
- Durchführung läuft „zeilenweise“ ab



```
MSGBOX «Geben Sie eine Zahl <5 ein»  
ELSE  
MSGBOX «Perfekt»  
END IF
```


Interpreter

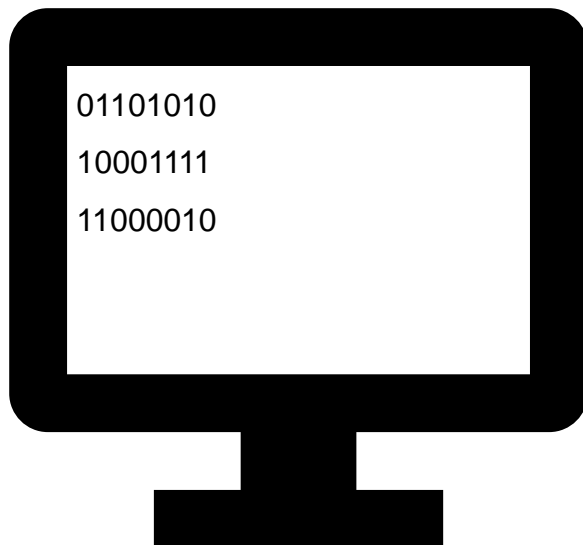
- Übersetzt einzelne Anweisungen in Maschinencode während Ausführung
- Durchführung läuft „zeilenweise“ ab



```
ELSE  
MSGBOX «Perfekt»  
END IF
```

Interpreter

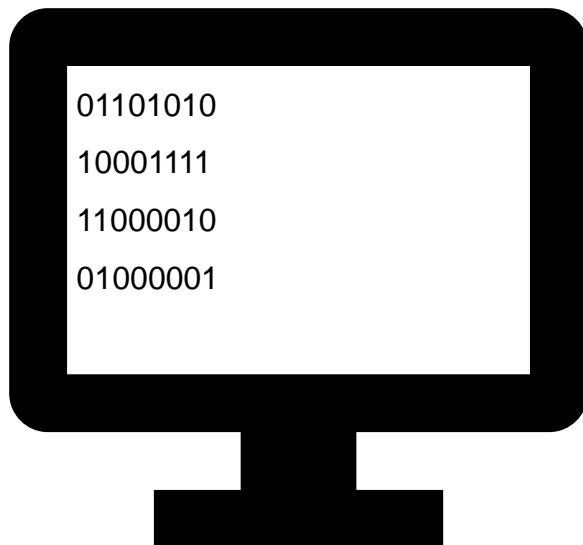
- Übersetzt einzelne Anweisungen in Maschinencode während Ausführung
- Durchführung läuft „zeilenweise“ ab



MSGBOX «Perfekt»
END IF

Interpreter

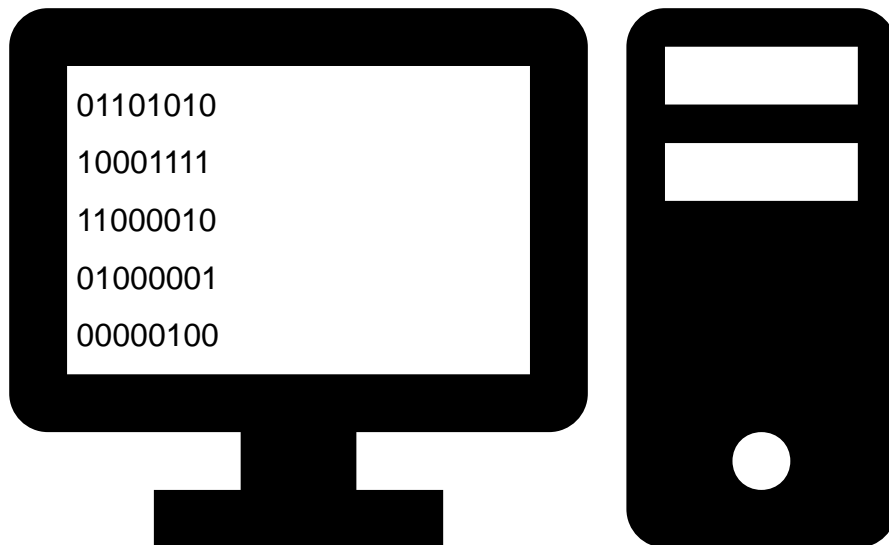
- Übersetzt einzelne Anweisungen in Maschinencode während Ausführung
- Durchführung läuft „zeilenweise“ ab



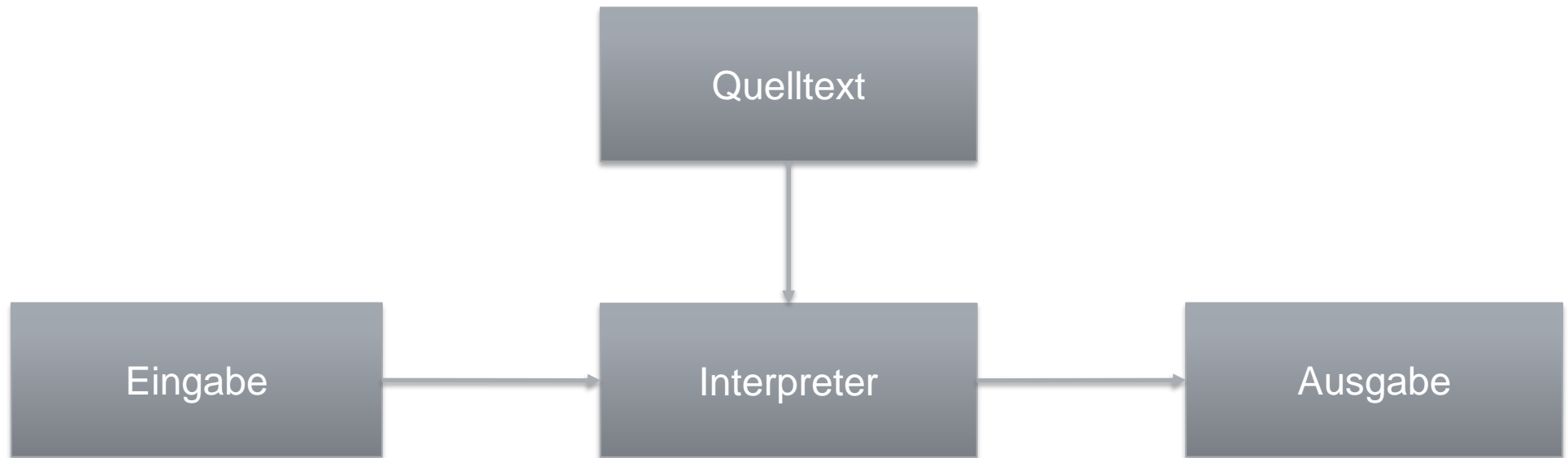
END IF

Interpreter

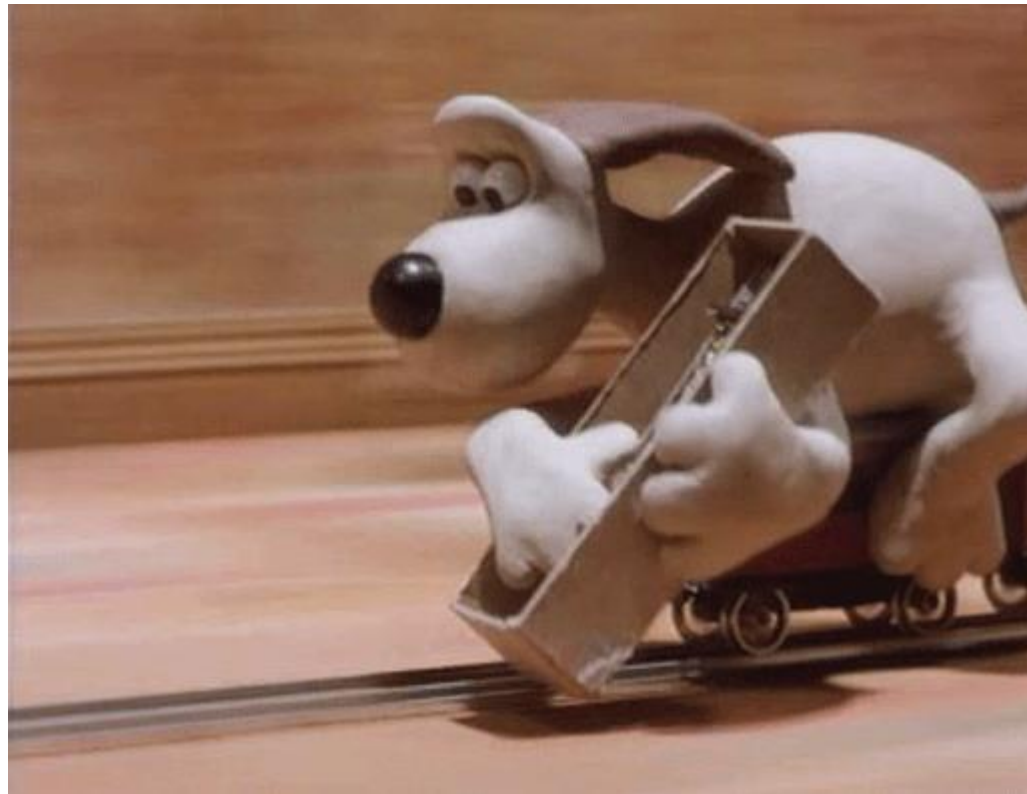
- Übersetzt einzelne Anweisungen in Maschinencode während Ausführung
- Durchführung läuft „zeilenweise“ ab



Interpreter

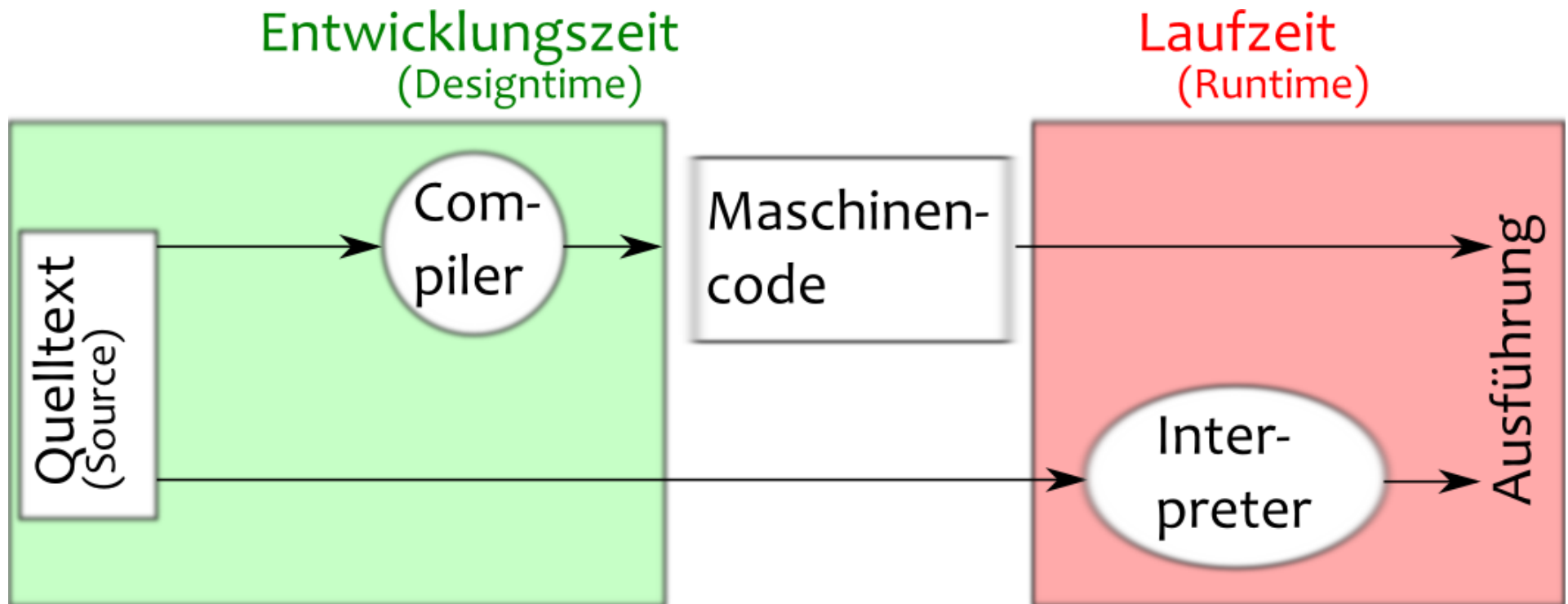


Interpreter



Quelle: <https://www.reddit.com/r/ProgrammerHumor/comments/3fizjs/jit/>
[Stand: 09.09.2022]

Compiler & Interpreter



Quelle: <http://www.roro-seiten.de/uhu/index.php?page=3972> [Stand: 09.09.2022]

Compiler & Interpreter

Vor- und Nachteile

Vorteile

Compiler

- Schnell bei Ausführung
- Syntaktische Fehler werden beim Kompilieren gefunden
- Compiler kann Code optimieren

Interpreter

- Kann sofort getestet werden
- Ausführbarer Code erst bei Laufzeit

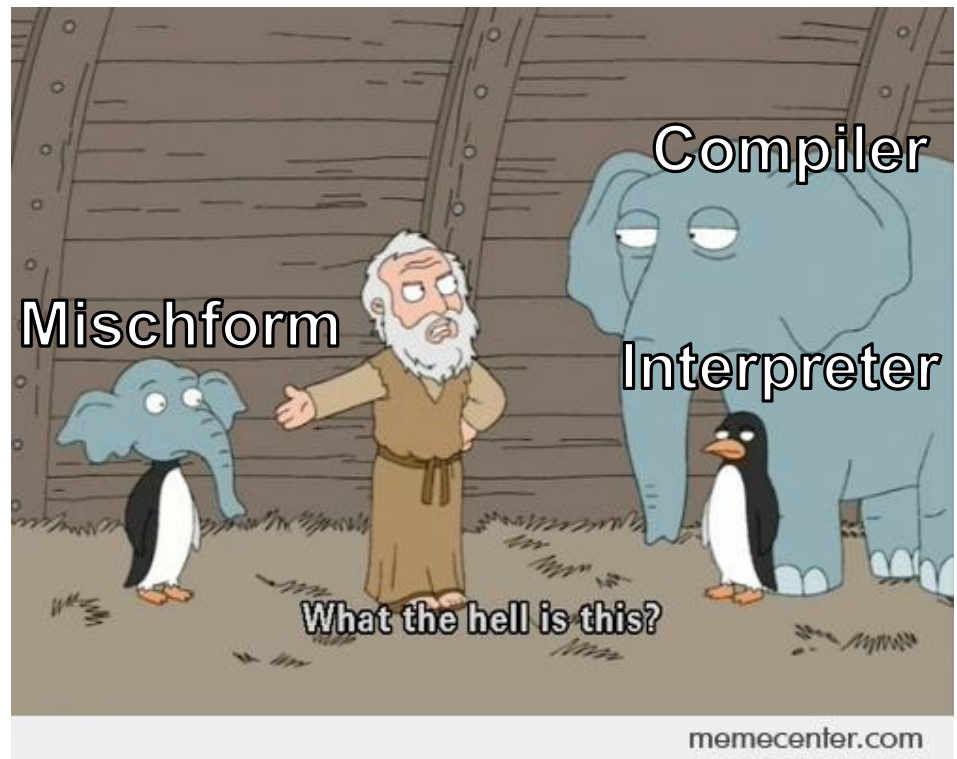
Nachteile

- Kompilieren benötigt Zeit und Ressourcen
- Änderungen im Quelltext müssen erneut kompiliert werden
- Auf andere Plattform muss erneut kompiliert werden

- Ausführung langsamer und ineffizienter
- Gleiche Programmteile müssen immer wieder übersetzt werden
- Fehler tritt bei Laufzeit auf. Programm lief bis dahin

Mischform

- Kombiniert Compiler mit Interpreter
 1. Quellcode wird in Bytecode kompiliert
 2. Bytecode wird zur Laufzeit interpretiert
- Interpretiertes Programm läuft auf einer virtuellen Maschine

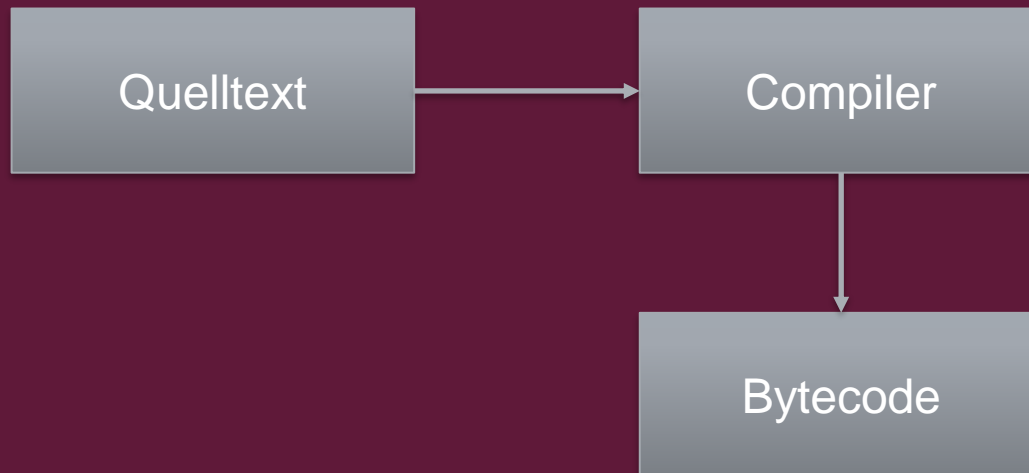


Quelle: <https://imgflip.com/memegenerator/167618150/Family-guy-Noahs-ark>
[Stand: 15.02.2022]. Bearbeitet

Mischform

Compiler + Interpreter

Übersetzungszeit (compile time)



Laufzeit (run time)



Compiler & Interpreter

Vor- und Nachteile

Vorteile

Compiler

- Schnell bei Ausführung
- Syntaktische Fehler werden beim Kompilieren gefunden
- Compiler kann Code optimieren

Interpreter

- Kann sofort getestet werden
- Ausführbarer Code erst bei Laufzeit

Mischform

- Plattformunabhängig
- Es können Programmteile kompiliert werden

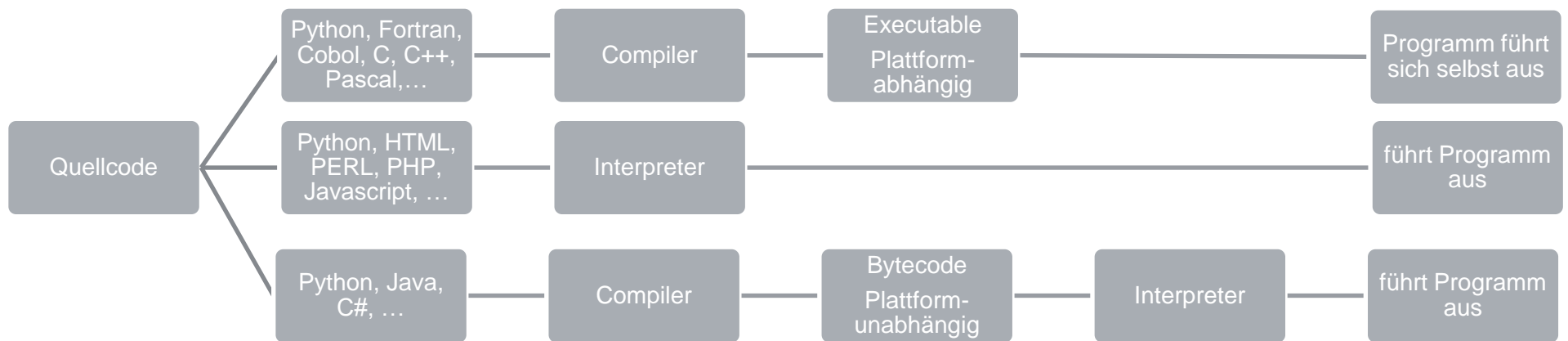
Nachteile

- Kompilieren benötigt Zeit und Ressourcen
- Änderungen im Quelltext müssen erneut kompiliert werden
- Auf andere Plattform muss erneut kompiliert werden

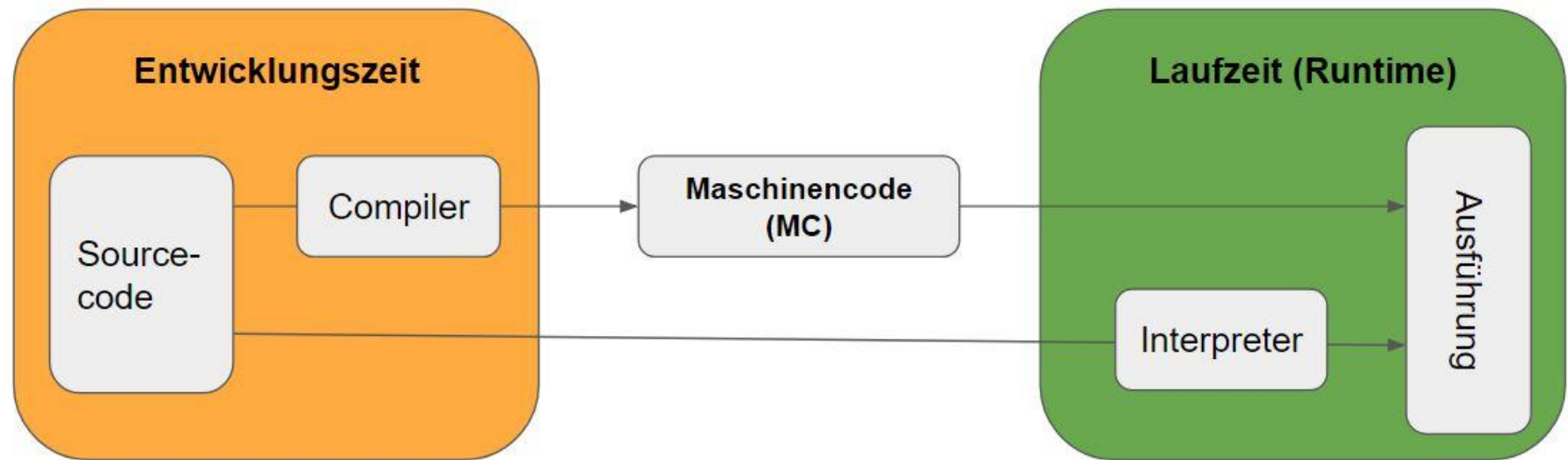
- Ausführung langsamer und ineffizienter
- Gleiche Programmteile müssen immer wieder übersetzt werden
- Fehler tritt bei Laufzeit auf. Programm lief bis dahin

- Programm läuft in einer Virtuellen Maschine

Zusammenfassung

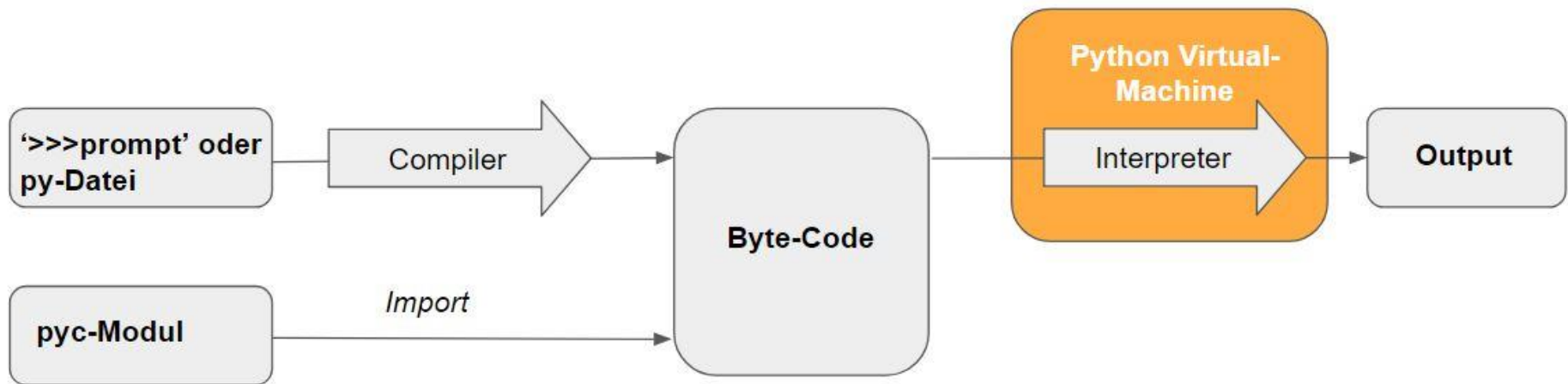


Vom Source-Code zum “Ergebnis”



Achtung gilt nur für CPython!

Vom Source-Code zum “Ergebnis”



Achtung gilt nur für CPython!

Kompilieren & Interpretieren

