Spielentwicklung in Python

Michael Finger

Gliederung

- Was ist Python
- Syntax
- TigerJython
- Snake

Was ist Python?

- Erstellt von Guido van Rossum in 1991
- Nachfolger der ABC Programmiersprache
- Momentan auf version 2.7 und 3.8
- Fokus auf Leserlichkeit

Warum Python

- Schnell zum schreiben
- Unkompliziert
- Interpretierte (nicht kompilierte Sprache)

Syntax

```
print "Hello, world!"
```

- Keine Semikolons
- Keine Klammern (print in 2.7)

Syntax

```
l = [1, 7, 6, "hi", 'hi']
```

- Liste nicht definiert auf einen Typen
- Länge nicht festgelegt

Syntax

```
d = {"same": "same", "yes": 1}
print(d.same)
```

- Dictionary == json datei
- Key:Value
- Aufruf durch d.key

Syntax

```
for i in range(10):
    print(i)
```

- Keine geschweiften Klammern
- For Loop wie for(int i: array)
- Doppelpunkt und Einrückung als seperatoren

Syntax

```
i = input("Eingabe: ")
if i < 7:
    print("Kleiner 7")
elif i > 12:
    print("Größer 12")
else:
    print("Keine Ahnung")
```

- Input() wie Scanner.readLine
- If, elif, else

Syntax

```
def printt(i, n):
    for i in range(n):
        print(i)

printt("Hi, ", 6)
```

- Funktionen mit def

Syntax

```
class Auto:
    def __init__(self):
        zünder = "Zünder"
        self.reifen = Reifen(6)

    def reifen_info(self):
        print("Reifengröße: %d", self.reifen.groesse)

class Reifen:
    def __init__(self, groesse):
        self.groesse = groesse
```

- Klasse: class
- Klassenvariablen: self.var_name = etwas
- Instanziierung ohne new
- Klassenfunktionen mit self
- Kostruktor = __init__

Syntax

```
import os
from multiprocessing import Process
from pathlib import *


class CustomPath(Path, Process):
    def __init__(self):
        os.system("ls")
```

- Imports mit import, Referenz auf diese Objekte mit dem Namen
- Imports mit from … import * oder Klasse, Referenz ohne Präfix
- Vererbung mit ()

TigerJython

- Lernumgebung für Python
- Angelehnt an Java (mit java.awt.Color etc)
- Entwickelt von Jake Arnold, Tobias Kohn und Aegidius Plüss
- Ziel:
  - Einfaches verwenden von Python im Lehrplan

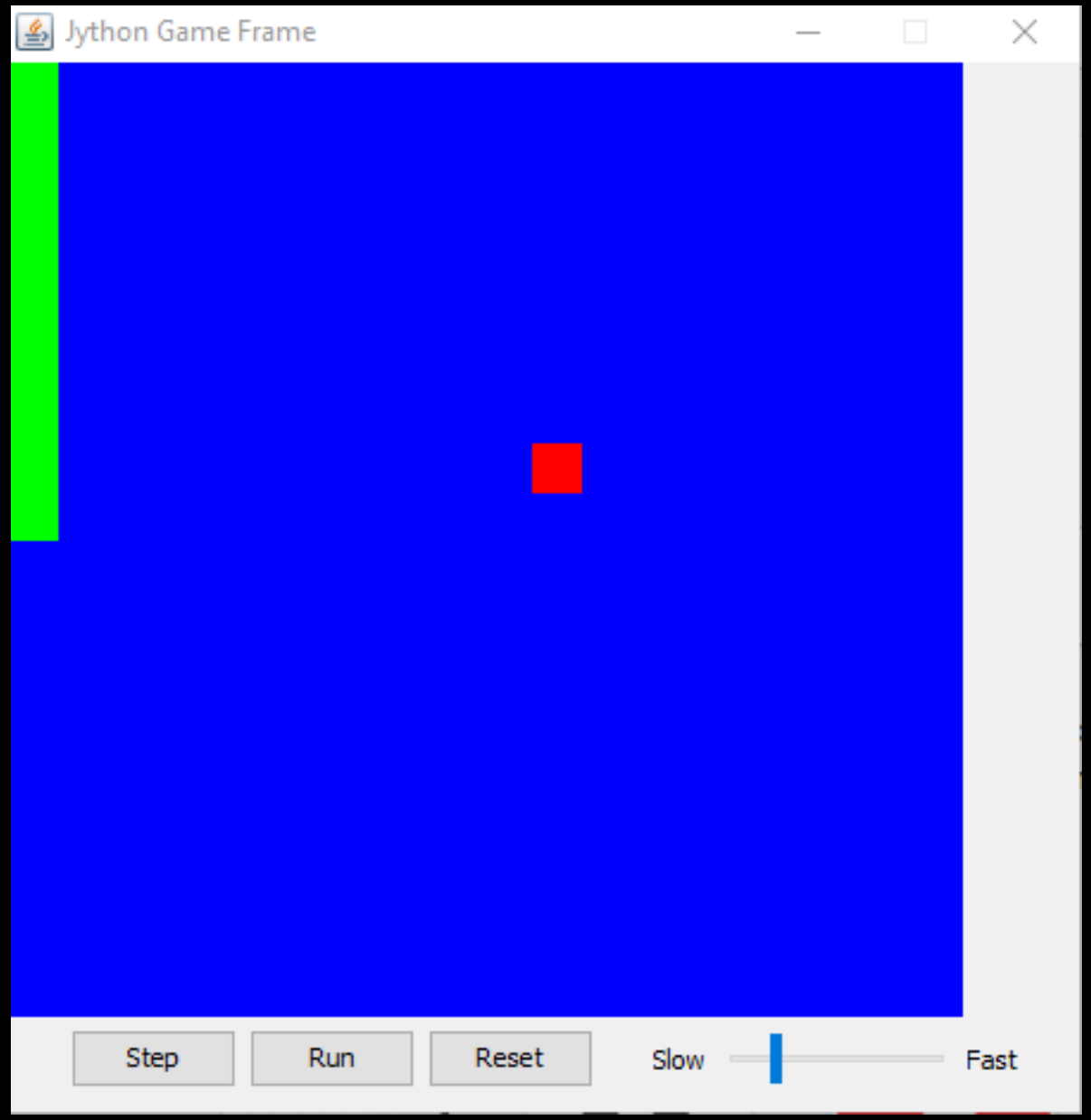
TigerJython

- Grundlage (für uns): GameGrid()
  - Simulationsgrundlage
  - Repräsentiert das Spielgitter
- Agierende Objekte: Actor
  - Assoziiert mit Sprite und Position
  - Jeden Simulationsschritt: Actor.act()

  →Gameobjekte müssen von Actor erben
- Position: Location:
  - X, Y auf dem Gamegrid

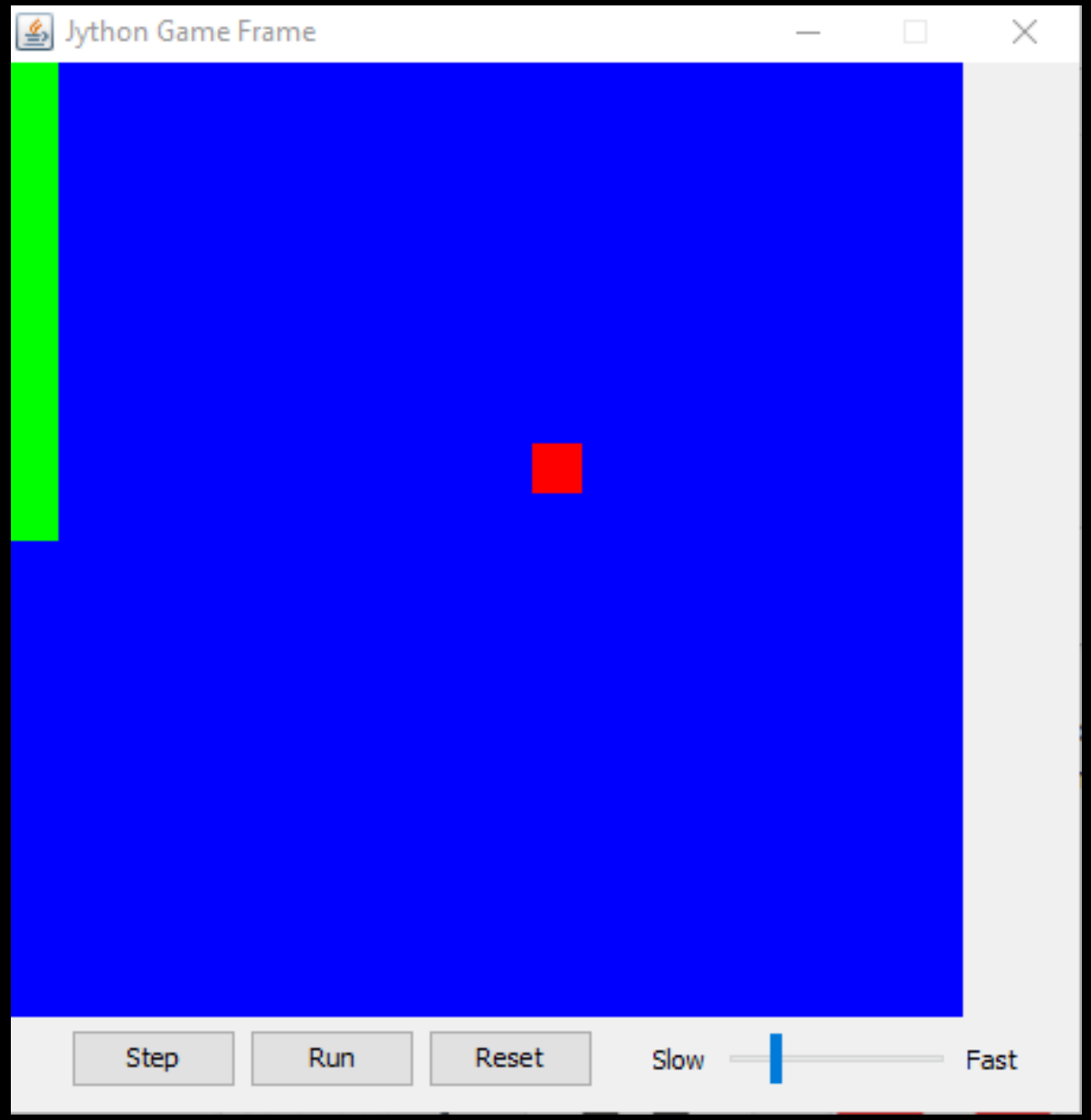
Snake

- Benötigte Spielobjekte:
  - GameGrid
  - Schlange
  - Essen



Snake

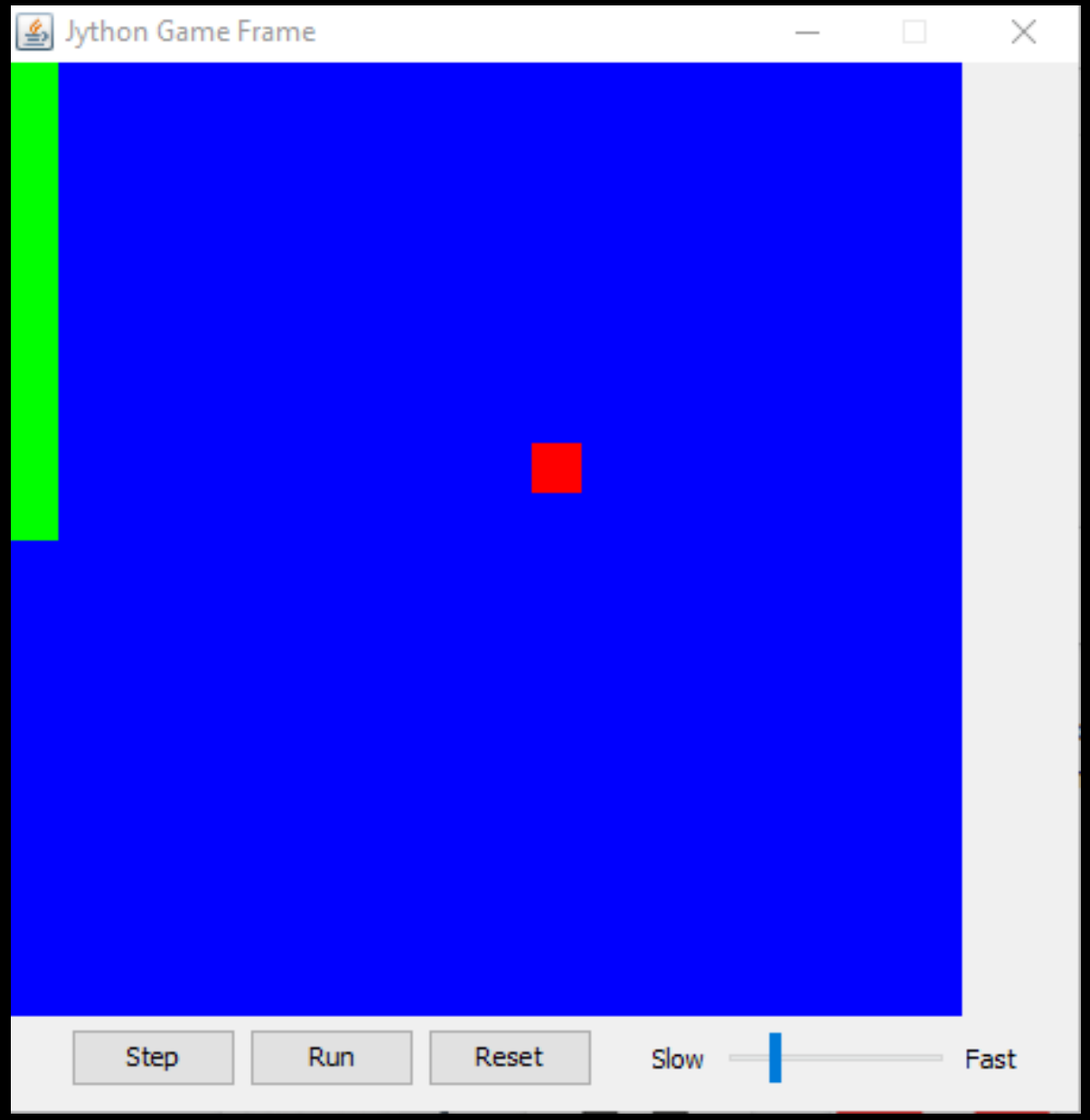
- Gamegrid:
  - Größe: 20*20 mit Zellengröße 20
  - Braucht show() und doRun() um Simulation zu starten
  - keyListener = function die auf Tastendrücke reagiert, siehe Javadocs für Keylistener



Snake

- Schlange:
  - Position (des Kopfes)
  - Richtung
  - Farbe
  - Länge
  - Schwanz (Positions[])

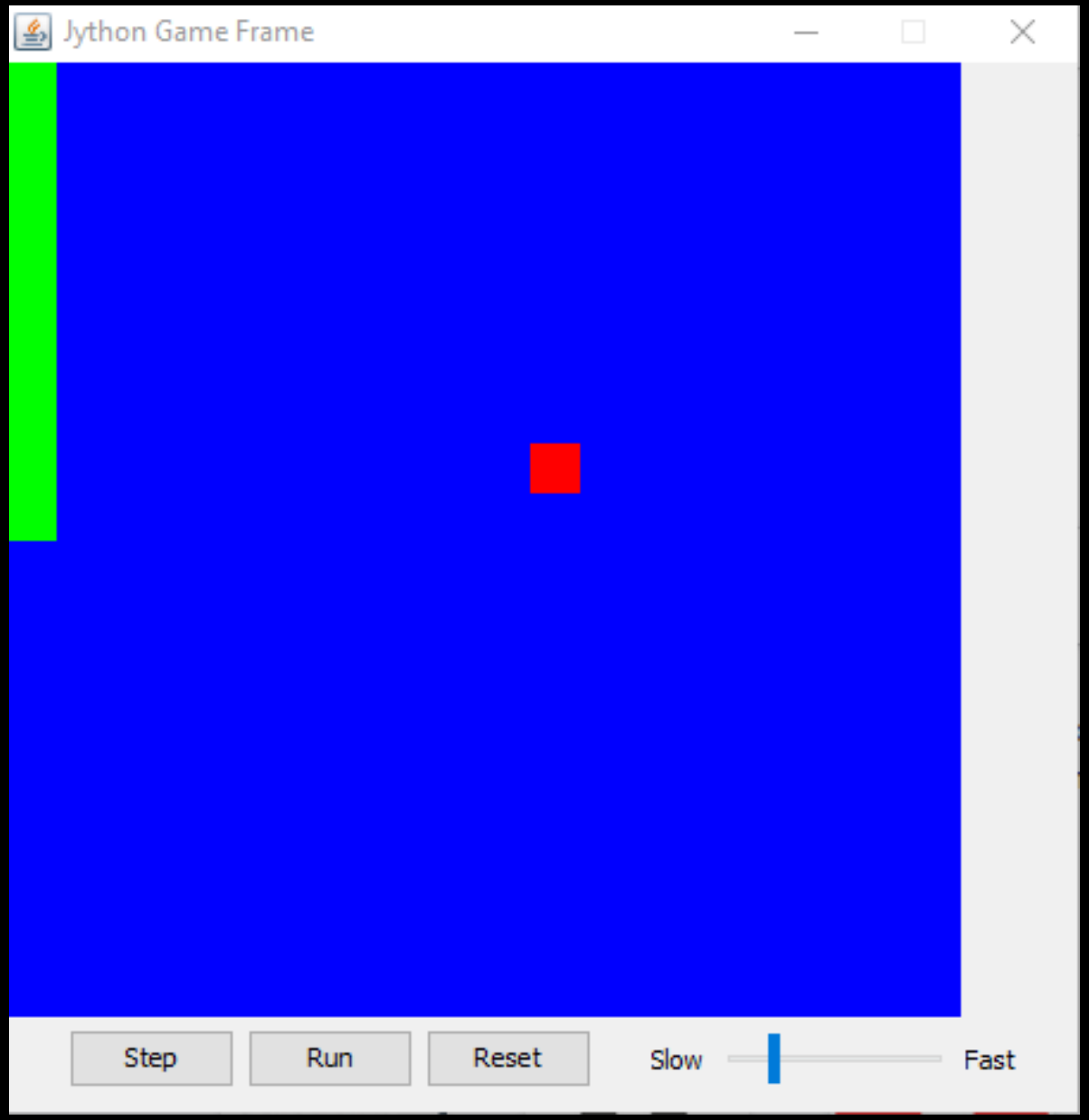
  - Act() (vorgegeben)
  - Bewegen()
  - Zeigen() (da wir keine Sprites Verwenden)
  - hatGegessen()
  - Kollidiert()



Snake

- Essen
  - Position
  - Farbe

  - Act()
  - wurdeGegessen()
  - neuePosition()
  - zeigen()



Snake

- Aufgabe: Programmiert ein simples Snake-Spiel mit der Lernumgebung TigerJython
- Links:
  - link.icytv.de/gfs/jython -- Downloadlink
  - link.icytv.de/gfs/jython-docs -- Dokumentation der Funktionen
- Klassen:
  - Schlange
  - Essen