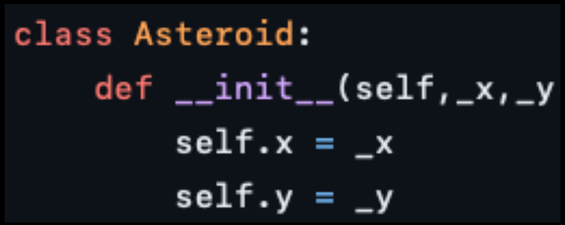
# Micro:bit – Asteroids Game

Model / View

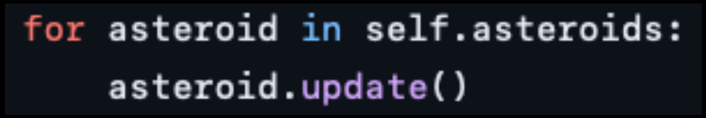
Andreas Schärer
KSR

# Zwei Arten von Code-Elementen in Game-Code

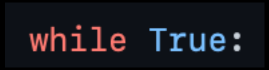
- **Normaler Python-Code:**
  - In jedem Python-Projekt
  - Auch ohne m:bit



```
class Asteroid:
    def __init__(self,_x,_y
        self.x = _x
        self.y = _y
```

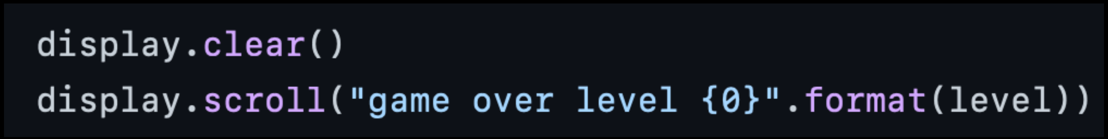


```
for asteroid in self.asteroids:
    asteroid.update()
```

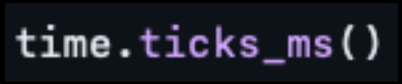


```
while True:
```

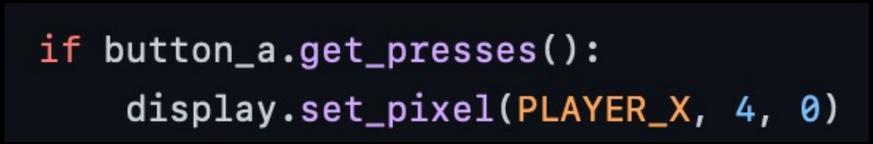
- **Spezifischer micro:bit-Code:**
  - Nur auf m:bit
  - Nicht in normalen Python-Projekt



```
display.clear()
display.scroll("game over level {0}".format(level))
```

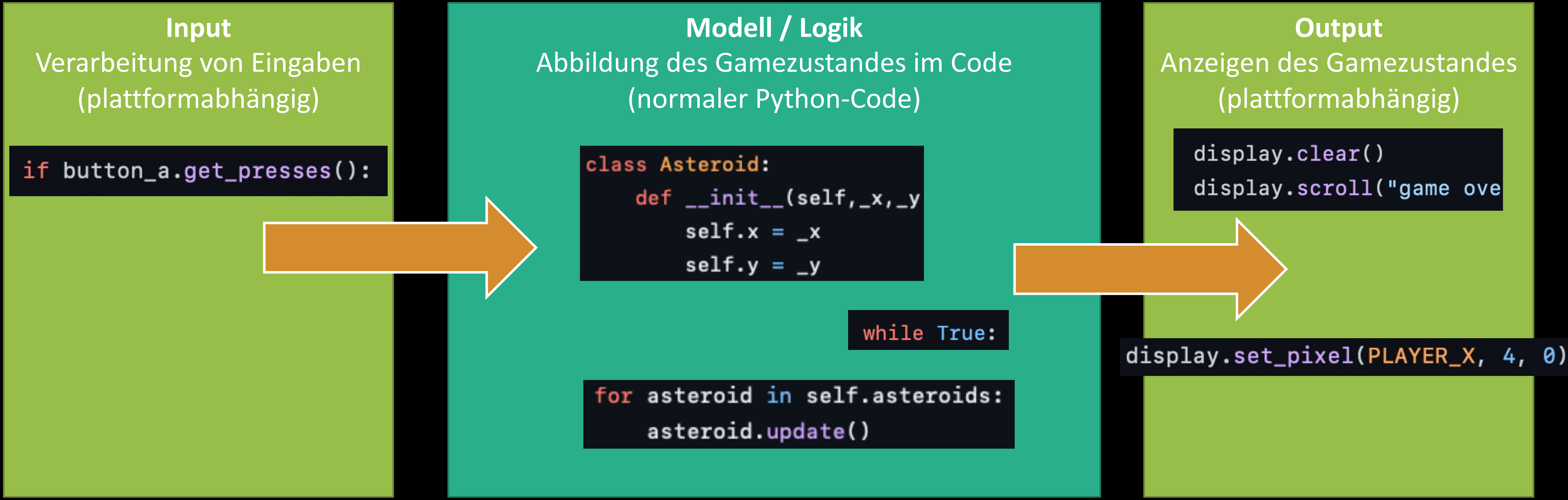


```
time.ticks_ms()
```



```
if button_a.get_presses():
    display.set_pixel(PLAYER_X, 4, 0)
```

# Zwei Arten von Code-Elementen in m:bit



**Input**
Verarbeitung von Eingaben
(plattformabhängig)

```
if button_a.get_presses():
```

**Modell / Logik**
Abbildung des Gamezustandes im Code
(normaler Python-Code)

```
class Asteroid:
    def __init__(self,_x,_y
        self.x = _x
        self.y = _y
```

```
while True:
```

```
for asteroid in self.asteroids:
    asteroid.update()
```

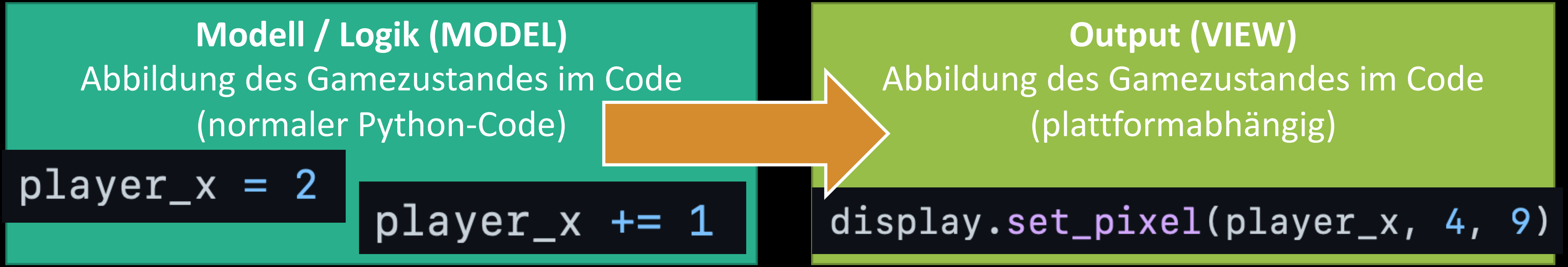
**Output**
Anzeigen des Gamezustandes
(plattformabhängig)

```
display.clear()
display.scroll("game ove
```

```
display.set_pixel(PLAYER_X, 4, 0)
```

# Model vs. View

- Programmier-Philosophie
- **Strikte Trennung Spiellogik (Model) und View (Input/Output):**
  - **Model**:
    - Bildet Zustand des Spiels abstrakt *im Code* ab
    - Nur normales Python (modulo ganz kleine Ausnahmen)
  - **View**:
    - Zeigt Spielzustand graphisch dar, übersetzt also Model in etwas, was angezeigt werden kann
    - Plattformspezifischer Code
- Beispiel:



**Modell / Logik (MODEL)**
Abbildung des Gamezustandes im Code
(normaler Python-Code)

```
player_x = 2
```

```
player_x += 1
```

**Output (VIEW)**
Abbildung des Gamezustandes im Code
(plattformabhängig)

```
display.set_pixel(player_x, 4, 9)
```

# Ziel

- **Asteroids Game: It’s the final Version!**
- Erstelle **Model** für Asteroids Game:
  - Neue Klasse Game, die alles vom Game beinhaltet
  - Beinhaltet gar nichts, was mit Input/Output (View) zu tun hat, wie:
    - `print(…)`
    - `display.set_pixel(…)`
    - `if button_a.get_presses():`

**View für Konsole**

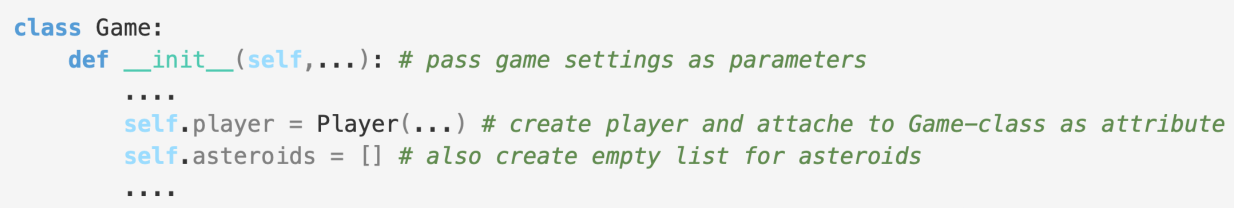
**View für microbit**

- Programmiere dann **drei verschiedene Views** für:
  1. Microbit
  2. Konsole
  3. Desktop App
  - **ALLE nutzen zu 100% den gleichen Model-Code!!!**

## Modell / Logik (MODEL)

Beinhaltet nur allgemeinen, gewöhnlichen Python-Code!

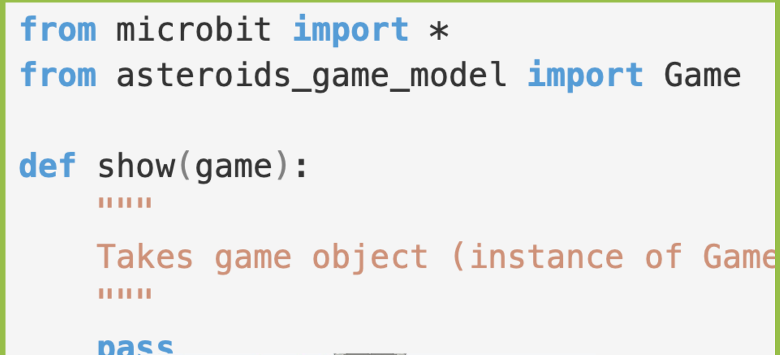
### asteroids_game_model.py



```
class Game:
    def __init__(self,...): # pass game settings as parameters
        ....
        self.player = Player(...) # create player and attache to Game-class as attribute
        self.asteroids = [] # also create empty list for asteroids
        ....
```

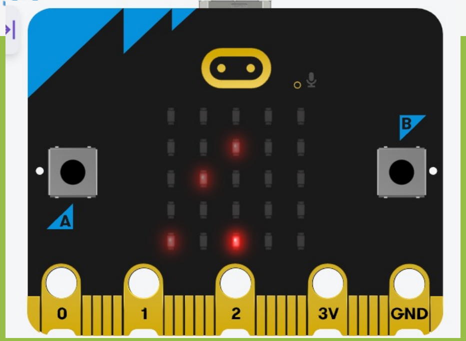
## VIEW (Input / Output)

### asteroids_game_microbit.py

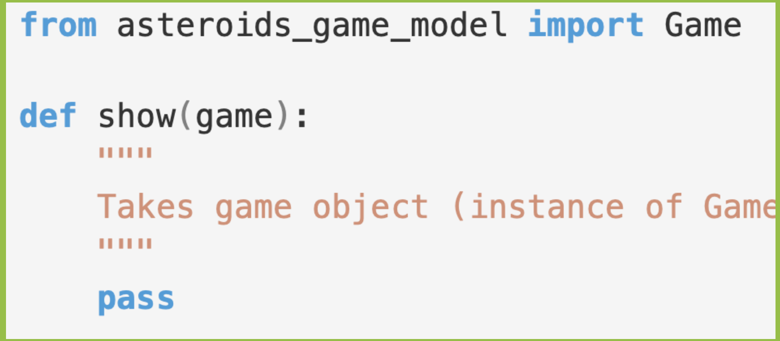


```
from microbit import *
from asteroids_game_model import Game

def show(game):
    """
    Takes game object (instance of Game
    """
    pass
```

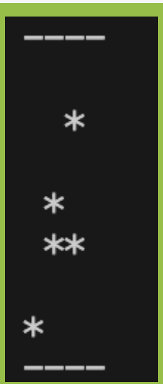


### asteroids_game_console.py



```
from asteroids_game_model import Game

def show(game):
    """
    Takes game object (instance of Game
    """
    pass
```



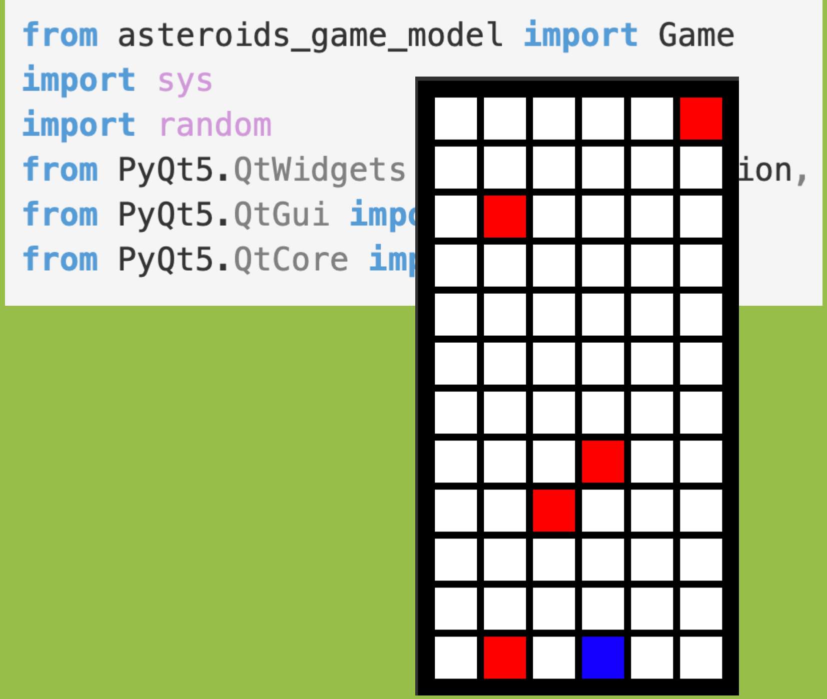
```
----

   *


  *
  **


*
----
```

### asteroids_game_pyqt.py



```
from asteroids_game_model import Game
import sys
import random
from PyQt5.QtWidgets                ion,
from PyQt5.QtGui impo
from PyQt5.QtCore im
```