py2cd Teil IV - Tastatur

Ein Spiel, bei dem der Benutzer nichts tun kann ist meistens ziemlich langweilig. Deswegen kann man auch mit py2cd Tastatureingaben abfangen und auf Mausbewegungen und Klicks reagieren.
Hinweis: Als Vorbereitung solltest du dich ein wenig mit Funktionen auskennen. Falls du dich damit noch nicht auskennst, hier eine kurze Info zu Funktionen.

> **Eine Funktion** ist ein Block von Code-Zeilen. Dieser Block bekommt einen eindeutigen Namen. Der Code-Block wird definiert, allerdings nicht sofort ausgeführt! Um diesen Block auszuführen, muss man lediglich dessen Namen gefolgt von runden Klammern im Quelltext verwenden.
> Ein Beispiel für eine Funktion ist die `print(..)`-Funktion. Diese Funktion ist vordefiniert und druckt nur dann etwas in der Konsole aus, wenn du sie aufrufst.
>
> Eine eigene Funktion kannst du so definieren:

```
def dein_name(): # hier definiert
    # Zeile 1
    # Zeile 2
    # ....

...
dein_name() # irgendwo später: hier aufgerufen
```

1.0.1 Wenn eine Taste gedrückt wird

Oft will man wissen ob eine bestimmte Taste gedrückt wird, z.B. die Leertaste, um eine Figur springen zu lassen. Diese Variante kann man als *'wenn gedrückt'*-Variante bezeichnet.
Um einen Tastendruck abzufangen, musst du eine Funktion definieren, die aufgerufen werden soll, wenn diese Taste gedrückt wird. Diese Funktion wird einmal aufgerufen, wenn die Taste nach unten gedrückt und einmal wenn diese Taste wieder losgelassen wird.
So wird eine Funktion für einen Tastendruck definiert:

```
def leer_gedrueckt(unten, pyg_ereignis):
    if unten:
        print("Leertaste gedrückt.")
```

Hierbei gibt der erste Parameter der Funktion an, ob die Taste gerade gedrückt (unten = Wahr/True) oder losgelassen wird (unten = Falsch/False).
Damit diese Funktion, auch bekannt ist, muss sie registriert werden:

```
# Tastendruck-Funktion registrieren
Spiel.registriere_taste_gedrueckt(T_LEER, leer_gedrueckt)
```

1.0.2 Solange eine Taste gedrückt ist

In vielen Fällen will meine Aktion ausführen, solange eine Taste gedrückt ist. Z.B. eine Spielfigur zu bewegen. Für diesen Fall gibt es eine andere Funktion. Diese Variante kann man als *'solange unten'*-Variante bezeichnet.
Dafür muss ebenfalls eine Funktion definiert werden, die solange wiederholt aufgerufen wird, wie die angegebenen Taste gedrückt gehalten wird.
So wird eine Funktion definiert, die für gedrückte Tasten aufgerufen wird:

```
def solage_rechts(dt):
    box.aendere_position(7 * dt, 0)
```

Damit diese Funktion, auch bekannt ist, muss sie registriert werden:

```
# Taste-Unten-Funktionen registrieren
Spiel.registriere_solange_taste_unten(T_RECHTS, solage_rechts)
```

Ein kleines Beispiel

Hier ein kleines Spiel, dass beide Arten von Tastendrücken empfängt. Zur Rechts-/Linksbewegung wird die *'solange unten'*-Variante verwendet und um einen Kreis zu schießen, wird die *'wenn gedrückt'*-Variante verwendet.

```
__author__ = 'Mark Weinreuter'

import random

from py2cd import *
from py2cd.farben import *

# Der erste Schritt, um ein Spiel zu starten ist immer, init()
 ↪ aufzurufen
Spiel.init(640, 480, "Mein Spiel")


def aktualisiere(dt):
    # alle Kreise durchgehen, bei Randberührung löschen -> in
 ↪ einer Liste speichern,
    # da aus einer for-Schleife nicht gelöscht werden darf!
    kreise_zu_loeschen = []

    # alle Blöcke bewegen
    for block in bloecke:
        block.bewege(dt)
```

```
        # Falls ein Block unsere Box berührt, ist das Spiel
 ↪  vorbei
        if block.beruehrt_objekt(box):
            # Spiel vorbei
            t = Text("Game Over", schrift=Schrift(50))
            t.zentriere()
            Spiel.entferne_aktualisierung()

        # Blöcke werden am unteren Rand zurücksetzen
        if block.beruehrt_unteren_rand():
            block.setze_position(random.randint(0, Spiel.breite
 ↪  - 20), 0)

    # alle Kreise bewegen und auf Kollisionen überprüfen
    for kreis in kreise:

        kreis.bewege(dt)
        for block in bloecke:
            if kreis.beruehrt_objekt(block):
                # Block zurücksetzen
                block.setze_position(random.randint(0,
 ↪  Spiel.breite - 20), 0)

        if kreis.beruehrt_oberen_rand():
            kreise_zu_loeschen.append(kreis)

    # Jetzt alle zu löschen gemerkten Kreise löschen
    for kreis in kreise_zu_loeschen:
        kreise.remove(kreis)
        kreis.selbst_entfernen()


# Diese Funktion wird aufgerufen, wenn die Leertaste gedrückt
 ↪  wird
def leer_gedrueckt(unten, pyg_ereignis):
    if unten:
        print("Leertaste gedrückt.")
        # Neuer Kreis. Setze dessen Mitte auf die Box-Mitte
        k = Kreis(0, 0, 10, BLAU)
        k.setze_geschwindigkeit(0, -8)
        k.mitte = box.mitte
        kreise.append(k)
    else:
        print("Leertaste losgelassen.")
```

```
# Aufgerufen solange die linke Pfeiltaste gedrückt ist
def solange_links(dt):
    box.aendere_position(-7 * dt, 0)


# Aufgerufen solange die rechte Pfeiltaste gedrückt ist
def solage_rechts(dt):
    box.aendere_position(7 * dt, 0)


def neuer_block():
    # Die x-Position und Fallgeschwindikeit zufällig wählen
    x = random.randint(0, Spiel.breite - 20)
    y_ges = random.randint(2, 4)

    block = Rechteck(x, 0, 20, 20, GRUEN)
    block.setze_geschwindigkeit(0, y_ges)
    bloecke.append(block)


bloecke = []
kreise = []

# ein gelbes Rechteckt, die Spielfigur
box = Rechteck(270, 200, 40, 40, ROT)
box.zentriere_horizontal()
box.unten = 25

for i in range(10):  # Zehn Blöcke generieren
    neuer_block()

# Tastendruck-Funktion registrieren
Spiel.registriere_taste_gedrueckt(T_LEER, leer_gedrueckt)
# Taste-Unten-Funktionen registrieren
Spiel.registriere_solange_taste_unten(T_RECHTS, solage_rechts)
Spiel.registriere_solange_taste_unten(T_LINKS, solange_links)

Spiel.zeichne_gitter()  # Hilfsgitter anzeigen
Spiel.setze_aktualisierung(aktualisiere)
Spiel.starten()
```