py2cd - Zeichnen mit Python Teil III

1.1 Vorbereitung

Um dieses Tutorial erfolgreich abzuschließen, muss zuerst pygame und py2cd installiert sein. Eine Installationsanleitung ist unter `https://github.com/coderdojoka/py2cd/` zu finden.
Desweiteren sollte Teil I und Teil II dieser Tutorial Serie verstanden sein.

1.2 Bewegung

Erinnerung: Der Koordinatenursprung `(0|0)` liegt in der linken oberen Ecke. Eine Änderung der x-Koordinate, die größer 0 ist, ist eine Bewegung nach rechts. Eine negative Änderung bedeutet eine Bewegung nach links.
Eine Änderung der y-Koordinate, die größer 0 ist, ist eine Bewegung nach **unten!**. Dies ist zunächst verwirrend, ist allerdings bei vielen Computerprogrammen so. Eine negative Änderung bedeutet demzufolge eine Bewegung nach oben!

```
box = Rechteck(10, 10, 50, 50, ROT)
# Ändert die x- und y-Koordinate um 5 Pixel
box.aendere_position(5, 5)
```

Dies ist also eine Bewegung um jeweils 5 Pixel nach rechts unten!

1.2.1 Objekte positionieren

Es gibt verschiedene Möglichkeiten Objekte, wie Kreise, Rechtecke, Bilder, etc.. zu platzieren:

Mit links, rechts, oben, unten:

Bei der Definition wird meistens die x-, y-Koordinate mit angegeben. Diese kann aber auch noch leicht nachträglich geändert werden, indem der Abstand zum linken, rechten, oberen oder unteren Rand angegeben wird.

```
# 1.) Am Anfang die linken obere Ecke angeben: 10, 10
box = Rechteck(10, 10, 50, 50, ROT)

# 2.) Mit links, rechts, oben, unten kann der Abstand zum jeweiligen Rand
# festgelegt werden, z.B. platziert:
# box.unten = 5 die Box-Unterkante 5 Pixel vom Boden entfernt
```

```
# 20 Pixel vom rechten Rand entfernt (ändert die x-Koordinate)
box.rechts = 20
# 20 Pixel vom linken Rand entfernt (ändert die x-Koordinate)
box.links = 20
# 20 Pixel vom oberen Rand entfernt (ändert die y-Koordinate)
box.oben = 20
# 20 Pixel vom unteren Rand entfernt (ändert die y-Koordinate)
box.unten = 20
```

Die Objekte-Mitte festlegen:

Man kann auch die Objektmitte direkt setzen:

```
# den Mittelpunkt des Objekts auf 20,30 setzen
box.mitte = (20, 30)
```

1.2.2 Die x-,y-Koordinaten ändern:

Relative Änderung

Die Position um einen Wert ändern. `x_neu = x_alt + wert` .

```
box = Rechteck(10, 10, 50, 50, ROT)
# Ändert die x- und y-Koordinate um 5 Pixel
box.aendere_position(5, 5)
```

Absolute Position setzen

Die Position auf einen Wert setzen. `x_neu = x_alt` .

```
box = Rechteck(10, 10, 50, 50, ROT)
# Setzt die x- und y-Koordinate auf die gegebenen Werte
box.setze_position(50, 50)
```

Hinweis: Diese Bewegungsmöglichkeiten sind ähnliche wie die in Scratch. `aendere_position` verhält sich genau wie `'ändere x um ...'` und `'ändere y um ...'` . `setze_position` verhält sich genau wie `'setze x auf ...'` und `'setze y auf ...'` .

1.2.3 Objekte zentrieren:

Objekte können horizontal, vertikal oder in beide Richtungen zentriert werden.

```
kreis = Kreis(10, 100, 100, BLAU)

# Zentriert das Objekt horizontal (ändert die x-Koordinate)
```

```
kreis.zentriere_horizontal()

# Zentriert das Objekt vertikal (ändert die y-Koordinate)
kreis.zentriere_horizontal()

# Zentriert das Objekt mittig  (ändert die x- und y-Koordinate)
kreis.zentriere()
```

1.2.4 Geschwindigkeit festlegen und bewegen

Man kann Objekte auch mithilfe von `bewege()` bewegen. Dabei wird das Objekt um die mit `setze_geschwindigkeit(6,6)` definierte Distanz (hier: 6 Pixel nach rechts und 6 Pixel nach unten).

```
box = Rechteck(10, 10, 50, 50, ROT)
# Die x- und y-Geschwindigkeit setzen
box.setze_geschwindigkeit(6,6)
# Bewegt die Box bei jedem Aufruf von bewege() um 6 Pixel
box.bewege()
```

1.2.5 Position abrufen

x- und y-Koordinaten

Man kann die x- und y-Koordinaten eines Objektes ganz einfach abfragen

```
# Box an der Stelle 10x10
box = Rechteck(10, 10, 50, 50, ROT)
# x- und y-Koordinate abfragen und ausgeben
print(box.x, box.y)
```

Abstand zum Rand

Genau wie man den Abstand zum linken, rechten, oberen oder unteren Rand setzen kann, kann man ihn auch abfragen. Das Gleiche gilt auch für die Objektmitte.

```
# Abstand zum linken Rand
abstand = box.links
print(abstand)

# Abstand zum rechten Rand
abstand = box.rechts
print(abstand)

# Abstand zum oberen Rand
abstand = box.oben
```

```
print(abstand)

# Abstand zum unteren Rand
abstand = box.unten
print(abstand)

# Mittelpunkt abfragen, ein Tupel mit (mitte_x, mitte_y)
mitte = box.mitte
print(mitte)
```

Größe des umgebenden Rechtecks

Wie zuvor beschrieben, ist jedes Objekt von einem unsichtbaren Rechteck eingegrenzt. Man kann auch dessen Breite und Höhe abfragen.

```
# Breite des umgebenden Rechtecks abfragen
breite = box.breite
print(breite)

# Höhe des umgebenden Rechtecks abfragen
hoehe = box.hoehe
print(hoehe)
```

1.3 Aufgaben

1. Bewege ein Objekt mit den verschiedenen Funktionen über die Spielfläche.
2. Frage die Position von Objekten und derren Abstand zum Rand ab. Ändert sich die Postion wie erwartet, wenn du sie bewegst? Z.B. sollte sich die x-Koordinate um 20 ändern, wenn `aendere_position(20, 0)` aufrufst.