Einfaches Spiel mit Python

Einführung in die Programmiersprache Python

BEZEICHNUNG:
Einfaches Spiel mit Python – Einführung in die Programmiersprache Python

DATUM:
27. Februar 2020

AUTOREN:
AdNovum Informatik AG

AdNovum

Inhaltsverzeichnis

1 Was ist eine Programmiersprache

1.1 Datenbehälter

In jeder Programmiersprache gibt es Datenbehälter oder **Variablen**. Diese sind, wie der Name schon verrät, Behälter, die Daten speichern können. Um eine solche Variable zu erstellen, muss man den Namen der Variable eingeben und den Wert, den man ihr zuweisen will:

```
number = 60
```

1.2 If

«if» ist ein Abfragungselement um Werte auf Wahr oder Falsch zu überprüfen. Die Überprüfung findet sehr oft zwischen zwei Feldern statt, also z.b « `30 == 40` ». Wichtig hierbei ist noch, dass für einen Vergleich wie diesen **zwei Gleichzeichen** benötigt werden, da ein Gleichzeichen bereits für Zuweisungen besetzt ist (*Oben wurde der Variable «number» mit einem Gleichzeichen den Wert 60 zugewiesen*).

Die Überprüfung beendet man mit einem Doppelpunkt, alles was danach kommt wird nur ausgeführt, **falls** die Überprüfung zutrifft.

```
number = 60


if number == 60:
   print("die Variable number ist 60")
else:
   print("die Variable number ist nicht 60")
```

Die Funktion «`print()`» gibt einen Text in die Konsole aus.

1.3 While Schlaufe

«while» ist dazu da, einen Teil des Codes so lange auszuführen, bis die Überprüfung nicht mehr zutrifft. Der Aufbau ist sehr ähnlich wie **«if»**.

```
number = 60
condition = True


while condition:
   number = number - 10
   if number <= 0:
      condition = False
```

Das Zeichen « **<** » steht für «**kleiner als**». Also ist «**<=**» «**Kleiner als oder gleich**». Das heisst, falls die Variable «`number`» kleiner als Null oder Null ist, soll die Variable «`condition`» zu Falsch geändert werden.

2 Pygame

Um ein Spiel mit Python zu programmieren, erstellen wir als erstes im «Games» Ordner ein neues Textdokument, benennen es **mit der Endung «**jump.py» und öffnen wir es mit Visual Studio Code. Jetzt sind wir bereit, um zu programmieren.

2.1 Erstellen eines Fensters

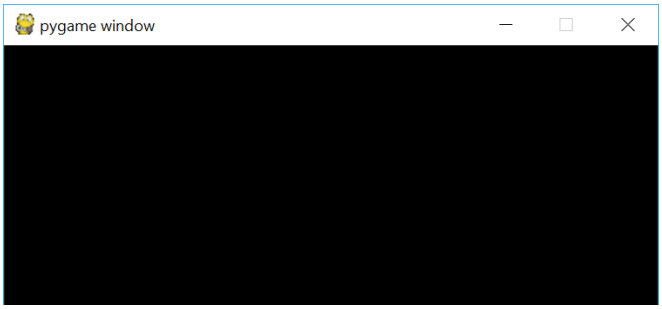
Nachdem wir das pygame importiert haben, ist es eine gute Idee, es zu initialisieren:

```
import pygame
pygame.init()
```

Wenn wir das getan haben, müssen wir ein Fenster einrichten, das unser Spiel darstellt.

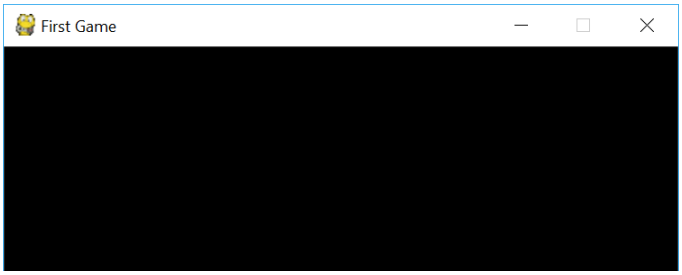
```
import pygame
pygame.init()


win = pygame.display.set_mode((500, 500)) # Dies erzeugt ein Fenster von 500 Breite und 500 Höhe.
```

Wenn wir nun das Programm ausführen ein Fenster, das so aussieht.



Das ist grossartig, aber wir möchten unserem Spiel einen kreativeren Namen geben als "pygame window". Dazu können wir Folgendes eingeben.

```
pygame.display.set_caption("First Game")
```



Das ist schon besser!

2.2 Bewegen eines Charakters

Wir werden damit beginnen, ein paar Variablen zu definieren, die unseren Charakter repräsentieren.

```
x = 50
y = 50
width = 40
height = 60
vel = 5
```

Jetzt werden wir unseren **Haupt- oder Spielkreislauf** einrichten. Alle Spiele haben eine Art Schleife, die ständig ausgeführt wird. Diese Schleife ist für Aufgaben wie die Überprüfung auf Ereignisse (wie Tastaturereignisse oder Kollisionen), das Bewegen von Objekten, die Aktualisierung des Displays und schliesslich das Beenden des Spiels zuständig.

In unserem Spiel werden wir eine "while" Schleife verwenden.

Innerhalb der Schleife werden wir eine Zeitverzögerung implementieren, damit wir die Geschwindigkeit des Spiels kontrollieren können. Wir werden auch damit beginnen, einige spezifische Ereignisse zu überprüfen.

```
import pygame
pygame.init()

win = pygame.display.set_mode((500,500))
pygame.display.set_caption("First Game")

x = 50
y = 50
width = 40
height = 60
vel = 5

run = True

while run:
    pygame.time.delay(100) # Dadurch wird das Spiel um die angegebene Anzahl von Millisekunden verzögert. In unserem Fall werden 0,1 Sekunden die Verzögerung sein.

    for event in pygame.event.get():  # Dabei wird eine Liste mit beliebigen Tastatur- oder Mausereignissen durchlaufen.

        if event.type == pygame.QUIT: # Überprüft, ob die rote Schaltfläche in der Ecke des Fensters angeklickt wurde
            run = False # Beendet die Spielschleife

pygame.quit() # Wenn wir die Schleife verlassen, wird dies ausgeführt und unser Spiel beendet.
```

Jetzt können wir ein Rechteck auf den Bildschirm zeichnen, um unseren Charakter darzustellen. Wir werden das Rechteck in der Hauptschleife so zeichnen, dass es bei jeder Schleife ständig neu gezeichnet wird.

```
    pygame.draw.rect(win, (255,0,0), (x, y, width, height))  # Diese nimmt: Fenster/Oberfläche, Farbe, Rechteck-Informationen
    pygame.display.update() # Dies aktualisiert das Fenster, so dass wir unser Rechteck sehen können.
```

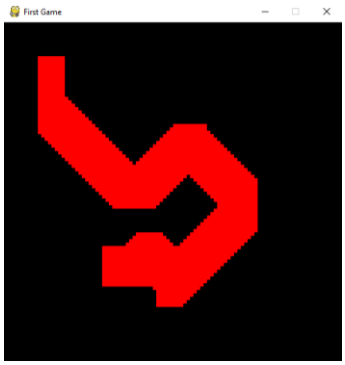
Nun können wir anfangen, nach Ereignissen zu prüfen, damit wir unseren Charakter bewegen können. Dies machen wir auch in der Hauptschleife.

```
    keys = pygame.key.get_pressed()  # Dadurch erhalten wir einen Array, in dem jede Taste einen Wert von 1 oder 0 hat, wobei 1 gedrückt und 0 nicht gedrückt ist.

    if keys[pygame.K_LEFT]: # So können wir prüfen, ob eine Taste gedrückt wird

    if keys[pygame.K_RIGHT]:

    if keys[pygame.K_UP]:

    if keys[pygame.K_DOWN]:
```

Innerhalb der if-Abfragen werden wir den Wert der Variablen x und y ändern, um den Charakter zu verschieben. Damit kommen wir zum **Koordinatensystem**. Im pygame ist die linke obere Ecke des Bildschirms (0,0) und die rechte untere Ecke (Breite, Höhe). Das heisst, um nach oben zu gehen, subtrahieren wir vom y unseres Charakters und um nach unten zu gehen, addieren wir zum y.

```
   if keys[pygame.K_LEFT]:
        x -= vel

    if keys[pygame.K_RIGHT]:
        x += vel

    if keys[pygame.K_UP]:
        y -= vel

    if keys[pygame.K_DOWN]:
        y += vel
```

Wenn wir nun das Programm ausführen und unser Rechteck bewegen, können wir immer noch alle vorherigen Rechtecke sehen.



Um dies zu beheben, müssen wir einfach über die vorherige Form zeichnen, bevor wir eine andere Form neu zeichnen. Wir können dazu `window.fill(color)` verwenden.

```
import pygame
pygame.init()

win = pygame.display.set_mode((500,500))
pygame.display.set_caption("First Game")

x = 50
y = 50
width = 40
height = 60
vel = 5

run = True

while run:
    pygame.time.delay(100)

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            run = False

    keys = pygame.key.get_pressed()

    if keys[pygame.K_LEFT]:
        x -= vel

    if keys[pygame.K_RIGHT]:
        x += vel

    if keys[pygame.K_UP]:
        y -= vel

    if keys[pygame.K_DOWN]:
        y += vel

    win.fill((0,0,0))  # Füllt den Hintergrund mit Schwarz
    pygame.draw.rect(win, (255,0,0), (x, y, width, height))
    pygame.display.update()

pygame.quit()
```

Jetzt können wir unseren Charakter bewegen!

2.3 Spielfeld festlegen

Im letzten Teil haben wir ein Rechteck erstellt, das wir mit den Pfeiltasten auf dem Bildschirm bewegen konnten. Wenn wir das Ende des Bildschirms erreicht haben, darf sich das Rechteck jedoch immer noch weiterbewegen. Um dies zu verhindern, müssen wir einige Grenzen festlegen und überprüfen, ob sich unser Rechteck innerhalb dieser Grenzen befindet, bevor wir es wieder bewegen.

Dazu können wir einfach die x- und y-Koordinaten des Rechtecks mit den Abmessungen des Bildschirms vergleichen. Wenn wir dies tun, müssen wir daran denken, dass wir, wenn wir etwas im pygame zeichnen, von der linken oberen Ecke des Objekts aus zeichnen. Das bedeutet, dass die linke obere Ecke unsere x- und y-Werte sind, während die rechte obere Ecke (x + Breite, y) und die linke untere Ecke (x, y + Höhe) ist.

```
    if keys[pygame.K_LEFT] and x > vel:  # Stellt sicher, dass die obere
linke Position unseres Charakters grösser ist als die Variable "vel", damit
wir uns nie vom Bildschirm entfernen.
        x -= vel

    if keys[pygame.K_RIGHT] and x < 500 - vel - width:  # Stellt sicher, dass
die obere rechte Ecke unseres Charakters kleiner ist als die Bildschirmbreite
minus seiner Breite.
        x += vel

    if keys[pygame.K_UP] and y > vel:  # Für die y-Koordinate gelten die
gleichen Prinzipien
        y -= vel

    if keys[pygame.K_DOWN] and y < 500 - height - vel:
        y += vel
```

Jetzt kann sich der Charakter nicht mehr vom Spielfeld entfernen!

2.4 Springen

Um das Springen zu ermöglichen, müssen wir ein paar Variablen einrichten.

```
# Dies geht fast an den Anfang des Programms, ausserhalb der while-Schleife.
isJump = False
jumpCount = 10
```

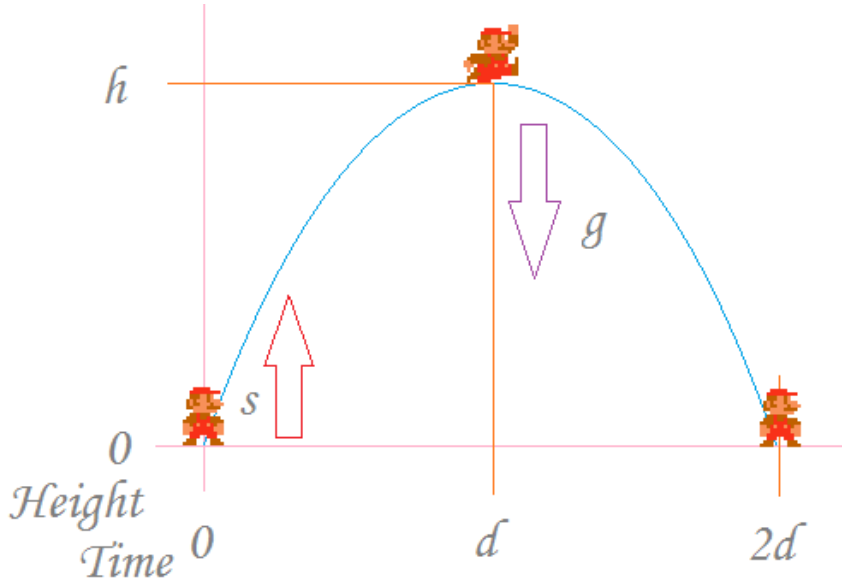
Jetzt werden wir prüfen, ob der Benutzer die Leertaste gedrückt hat. Wenn dies der Fall ist, ändern wir «isJump» auf «True» und starten zu springen.

```
# Befindet sich in der while-Schleife, unter dem Event zum Abwärts bewegen.
if keys[pygame.K_SPACE]:
    isJump = True
```

Wenn wir jetzt springen, wollen wir nicht zulassen, dass der Benutzer wieder springt oder sich auf und ab bewegen kann. Um dem Benutzer dies zu verbieten, werden wir eine weitere if/else-Anweisung hinzufügen.

```
if not(isJump): # Prüft, ob der Benutzer nicht springt
        if keys[pygame.K_UP] and y > vel:
            y -= vel

        if keys[pygame.K_DOWN] and y < 500 - height - vel:
            y += vel

        if keys[pygame.K_SPACE]:
            isJump = True
    else:
        # Das wird passieren, wenn wir springen
```

Wir werden jetzt mit dem Schreiben des Codes anfangen, um den Charakter springen zu lassen. Wir setzen eine quadratische Formel für unser Springen um. Denn im Idealfall möchten wir, dass unser Sprung glatt ist und ungefähr so aussieht:



Innerhalb des «else» werden wir folgendes einfügen.

```
if jumpCount >= -10:
    y -= (jumpCount * abs(jumpCount)) * 0.5
    jumpCount -= 1
else: # Dies wird ausgeführt, wenn der Sprung beendet ist.
    jumpCount = 10
    isJump = False
    # Zurücksetzen unserer Variablen
```

Und jetzt kann unser Charakter springen!

2.5 Vollständiger Code

Hier steht, wie der vollständige Code aussehen sollte.

```
import pygame
pygame.init()

win = pygame.display.set_mode((500,500))
pygame.display.set_caption("First Game")

x = 50
y = 50
width = 40
height = 60
vel = 5

isJump = False
jumpCount = 10

run = True

while run:
    pygame.time.delay(100)

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            run = False

    keys = pygame.key.get_pressed()

    if keys[pygame.K_LEFT] and x > vel:
        x -= vel

    if keys[pygame.K_RIGHT] and x < 500 - vel - width:
        x += vel

    if not(isJump):
        if keys[pygame.K_UP] and y > vel:
            y -= vel

        if keys[pygame.K_DOWN] and y < 500 - height - vel:
            y += vel

        if keys[pygame.K_SPACE]:
            isJump = True
    else:
        if jumpCount >= -10:
            y -= (jumpCount * abs(jumpCount)) * 0.5
            jumpCount -= 1
        else:
            jumpCount = 10
            isJump = False

    win.fill((0,0,0))
    pygame.draw.rect(win, (255,0,0), (x, y, width, height))
    pygame.display.update()

pygame.quit()
```