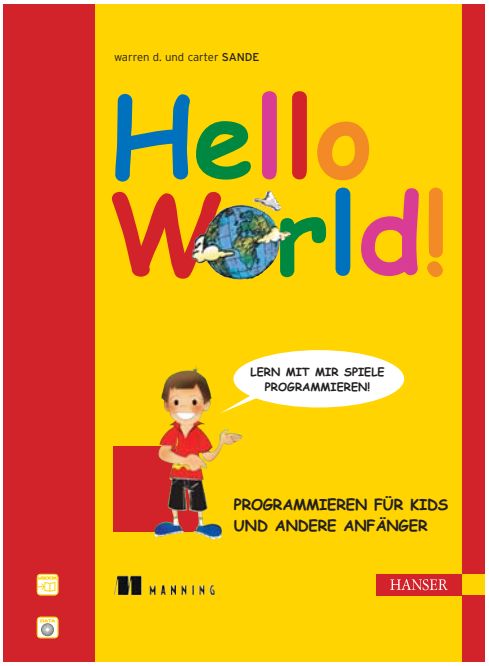
HANSER



Erratum

Warren D. und Carter Sande

"Hello World"

Programmieren für Kids und andere Anfänger

ISBN 978-3-446-42144-8

Weitere Informationen oder Bestellungen unter
http://www.hanser.de/978-3-446-42144-8
sowie im Buchhandel

Hallo liebe Leser von „Hello World!“,

leider haben sich im Buch ein paar Fehler eingeschlichen, und zwar auf den Seiten 49, 60, 99 und 100. Auf dieser und den nächsten Seiten haben wir den Text korrigiert. (In den Quellcodes auf der CD stimmt alles.)

```
print "Das sind",
print celsius,
print "Grad Celsius"
```

Beachte die Kommas am Ende dieser Zeilen

Du kannst auch die letzten drei Zeilen von Listing 5.3 zu einer Zeile zusammenfassen, wie hier:

```
print "Das sind", celsius,"Grad Celsius"
```

Das ist nur eine Abkürzung für die drei **print**-Anweisungen, die wir zuvor hatten.

int() mit raw

Diese Zeile muss heißen:
`anzahl = int(antwort)`

Wenn die Zahl, die … (also keine Dezimalzahl) ist, kannst du sie mit **int** … konvertieren:

```
antwort = raw_input("Wie viele Schüler sind in deiner Klasse: ")
anzahl = int(response)
```

Selbst wenn der Benutzer einen Float eingibt, bekommt er diesen abgerundet als Integer zurück.

Denken wie ein (Python)-Programmierer

Es gibt noch eine andere Möglichkeit, Zahlen als Eingabe zu bekommen. Python hat eine Funktion namens **input()**, die dir direkt Zahlen gibt, ohne dass du sie mit **int()** oder **float()** konvertieren musst. Wir haben sie beim Zahlen raten in Kapitel 1 schon benutzt, weil sie der einfachste Weg ist, um eine Zahl von einem Benutzer zu bekommen.

Es gibt jedoch einige Gründe, die gegen **input()** sprechen. Einer ist, dass diese Funktion aus künftigen Python-Versionen entfernt wird (Versionen 3.0 und höher). Dann gibt es nur noch die Funktion **raw_input()**, das zwar anschließend in **input()** umbenannt wird, aber immer die Funktion bleibt, die du in diesem Kapitel gesehen hast, und die nur Strings nimmt.

Da wir wissen, wie wir aus einem String eine Zahl machen können, empfehle ich dir, statt **input()** besser **raw_input()** zu verwenden.

Übrigens wird Python 3.0 auch noch eine andere Änderung bringen: Anstatt
print "Hallo"
wirst du dort schreiben müssen:
print ("Hallo")
Ab Python3.0 werden die runden Klammern für print erforderlich sein.

EasyGui hat auch eine `integerbox`, die du für die Eingabe von Integern verwenden kannst. Du kannst eine Ober- und eine Untergrenze für die Zahl festlegen, die eingegeben wird.

Diese Box lässt dich allerdings keine Floats (Dezimalzahlen) eingeben. Um das zu tun, müsstest du eine Enterbox einsetzen und den String dann mit `float()` umwandeln.

Nochmal Zahlen raten

In Kapitel 1 schrieben wir ein einfaches Spielprogramm, „Zahlen raten". Jetzt wollen wir dasselbe noch einmal tun, aber dieses Mal EasyGui für die Ein- und Ausgabe verwenden. Listing 6.5 zeigt den Code.

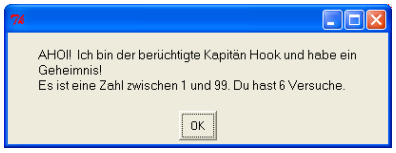
Listing 6.5 Zahlen raten mit EasyGui

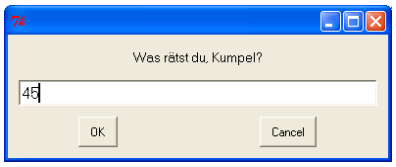
```
import random, easygui

geheimnis = random.randint(1, 99)
tipp = 0
versuche = 0

easygui.msgbox("""
habe ein Geheimnis
Versuche.""")

while tipp != geheimnis und versuche < 6:
    tipp = easygui.integerbox("Was rätst du, Kumpel?")
    if not tipp: break
    if tipp < geheimnis:
        easygui.msgbox(str(tipp) + " ist zu niedrig, du Landratte!")
    elif tipp > geheimnis:
        easygui.msgbox(str(tipp) + " ist zu hoch, du Leichtmatrose!")
    versuche = versuche + 1

if tipp == geheimnis:
    easygui.msgbox("Ha! Du hast es! Hast mein Geheimnis erraten!")
else:
    easygui.msgbox("Alle Versuche verbraucht! Mehr Glück beim " +\
                   "nächsten Mal!, Kumpel!")
```

Diese Zeile muss heißen:

```
while tipp != geheimnis and versuche < 6:
```

Wir haben immer noch nicht gelernt, wie alle die Teile dieses Programms funktionieren, aber du kannst es einfach eintippen und ausprobieren. Es müsste diese Ausgabe haben:





```
def aktualisiereHindernisGruppe(karte0, karte1):
    hindernisse = pygame.sprite.Group()
    for ob in karte0:  hindernisse.add(ob)
    for ob in karte1:  hindernisse.add(ob)
    return hindernisse

pygame.init()
screen = pygame.display.set_mode([640,640])
uhr = pygame.time.Clock()
skier = SkierKlasse()
geschwindigkeit = [0, 6]
karten_position = 0
punkte = 0
karte0 = erstelle_karte(20, 29)
karte1 = erstelle_karte(10, 19)
aktive_karte = 0
hindernisse = aktualisiereHindernisGruppe(karte0, karte1)
font = pygame.font.Font(None, 50)

while True:
    uhr.tick(30)
    for event in pygame.event.get():
        if event.type == pygame.QUIT: sys.exit()
        if event.type == pygame.KEYDOWN:
            if event.key == pygame.K_LEFT:
                geschwindigkeit = skier.wende(-1)

        on >=640 and aktive_karte == 0:
        akti    1
        karte0 = er   e_karte(20, 29)
        hindernisse = updateHindernisGroup(karte0, karte1)
    if karten_position >=1280 and aktive_karte == 1:
        aktive_karte = 0
        for ob in karte0:
            ob.ort[1] = ob.ort[1] -
        karten_position = karten_positi
        karte1 = erstelle_karte(10, 19)
        hindernisse = aktualisierenHindernisGruppe(karte0, karte1)

    for hindernis in hindernisse:
        hindernis.scrolle(karten_position)
```

geht zum nächsten Landschaftsbild

macht alles bereit

startet Hauptschleife

aktualisiert die Grafik 30 Mal pro Sekunde

achtet auf Tastendrücke oder Fenster schließen-Befehl

wechselt

Diese Zeile muss heißen:
`hindernisse = aktualisiereHindernisGruppe(karte0, karte1)`

Statt `aktualisieren` **muss es in dieser Zeile heißen:** `aktualisiere`

Statt `type` muss es in dieser Zeile heißen: `typ`

```
        getroffen =  pygame.sprite.sprit                (skier, hindernisse, False)
        if getroffen:
            if getroffen[0].type == "baum" and not getroffen[0].vorbei:
                punkte = punkte - 100
                skier.image = pygame.image.load("skier_crash.png")
                animiere()
                pygame.time.delay(1000)
                skier.image = pygame.image.load("skier_runter.png")
                skier.winkel = 0
                geschwindigkeit = [0, 6]
                getroffen[0].vorbei = True
            elif getroffen[0].type == "fahne" and not getroffen[0].vorbei:
                punkte += 10
                hindernisse.remove(ge          [0])

        punktzahl_text = font.re
        animiere()
```

prüft, ob Bäume umgefahren oder Fahnen gesammelt werden

Auch hier muss es statt `type` heißen: `typ`

Der Code von Listing 10.1 liegt im Ordner **\Beispiele\skier**. Wenn du also steckenbleibst oder nicht das alles eintippen möchtest, kannst du auch die Datei benutzen. Aber glaube mir: Durch Eintippen lernst du mehr, als wenn du nur eine Datei öffnest und das Listing anschaust.

In späteren Kapiteln werden wir alle Schlüsselwörter und Techniken kennenlernen, die in Skier verwendet werden. Vorläufig tippe es einfach ein und versuche es.

0011000111001110000110110100011011010111001100011001100110100110011100100110

Probiere es aus

1 Das Einzige, was du in diesem Kapitel tun sollst, ist das Skier-Programm einzugeben (Listing 10.1) und auszuprobieren. Wenn bei der Ausführung eine Fehlermeldung auftritt, schau sie dir genau an und versuche herauszufinden, wo der Fehler steckt.

   Viel Glück!