HOPP ITs

CODE MAINIA

Python

Kartenbeschreibung

KARTENBESCHREIBUNG

Hellgrüne Karten: WERTZUWEISUNGEN UND OPERATOREN	BEDEUTUNG
zahl = 7	Weist der Variablen »zahl« den Wert 7 zu.
zahl = zahl + 4	Erhöht den Wert der Variablen »zahl« um 4.
zahl = zahl – 2	Verringert den Wert der Variablen »zahl« um 2.
zahl = zahl * 3	Multipliziert den Wert der Variablen »zahl« mit 3.
zahl = 1 + 3 * zahl	Es gilt die Punkt-vor-Strich-Rechnung. Multipliziert den Wert der Variablen »zahl« erst mit 3 und erhöht den neuen Wert anschließend um 1.

Grüne Karten: VERWIRRUNGSKARTEN UND VERGLEICHSOPERATOREN	BEDEUTUNG
zahl == 5	Prüft, ob die Variable »zahl« den Wert 5 hat und gibt einen booleschen Wert (wahr/falsch) zurück. Der Wert der Variablen »zahl« bleibt unverändert. Weitere Vergleichsoperatoren: <, <=, >, >=, != (ungleich).
zahl >= 5 zahl < 12	Der boolesche Wert verändert den Wert der Variablen »zahl« nicht.
x = 5 ZahL = 5 zah = 5	Eine neue Variable wird deklariert und initialisiert. Sie verändert den Wert der Variablen »zahl« nicht.
zahl = zahl – zahl	Da der alte Wert der Variablen »zahl« von sich selbst subtrahiert wird, ändert sich hier der Wert der Variablen »zahl« immer zu Null.
zahl = 5 zahl = zahl + 2*zahl	Bei mehrzeiligen Anweisungen werden die Zeilen nacheinander ausgeführt. Im gezeigten Beispiel hat die Variable »zahl« unabhängig von ihrem vorherigen Wert nach dieser Karte den Wert 15.

Dunkelgrüne Karten: GANZZAHLIGE DIVISION MIT REST

	BEDEUTUNG
zahl = zahl // 4	Dividiert den Wert der Variablen »zahl« durch 4. Beachte: Falls bei der Division mit // Nachkommastellen entstehen, werden diese abgeschnitten, z. B. 15 // 4 = 3, denn 15 : 4 = **3**,75.
zahl = zahl % 2	Weist der Variablen »zahl« den Rest beim Teilen des Werts durch 2 zu. Beispiel: 7 : 2 = 3 Rest 1, also 7 % 2 = 1.
wenn »zahl« negativ:	–8 : 5 = –1 Rest –3, also -8 % 5 = –3. Alternativ kann immer eine nichtnegative Zahl gewählt werden, indem man zunächst das kleinstmögliche Vielfache des Divisors addiert (im Beispiel: +10), sodass ein nichtnegativer Wert entsteht: –8 + 10 = 2, also –8 % 5 = 2.

Gelbe Karten: BEDINGTE ANWEISUNG UND VERZWEIGUNG

	BEDEUTUNG
if zahl <= 4: Anweisung	Führt die nachfolgende eingerückte Anweisung nur dann aus, wenn die Bedingung »zahl <= 4« erfüllt (wahr) ist.

if zahl > 7:
  Anweisung 1
else:
  Anweisung 2
	Führt die Anweisung 1 nur dann aus, wenn die Bedingung »zahl > 7« erfüllt ist. *Ansonsten* wird Anweisung 2 ausgeführt.

if zahl < 5:
  Anweisung 1
elif zahl > 10:
  Anweisung 2
else:
  Anweisung 3
	Es wird in allen Fällen *entweder* Anweisung 1 *oder* Anweisung 2 *oder* Anweisung 3 ausgeführt, also immer genau eine der drei Anweisungen.

if zahl < 0:
  zahl = zahl + 1
if zahl >= 0:
  zahl = zahl – 1
	Zwei bedingte Anweisungen werden *nacheinander* ausgeführt. Hat »zahl« z. B. den Wert –1, so wird ihr Wert um 1 erhöht und wieder um 1 verringert, sodass der Wert der Variablen »zahl« unverändert bleibt.

if zahl > 4:
  zahl = zahl + 5
if zahl > 20:
  zahl = zahl – 10
else:
  zahl = zahl + 10
	Eine bedingte Anweisung und eine Verzweigung werden *nacheinander* ausgeführt. Hat »zahl« z. B. den Wert 16, so wird ihr Wert um 5 erhöht und dann wieder um 10 verringert.

Orange Karten:
IF-ANWEISUNGEN MIT GANZZAHLIGER DIVISION

	BEDEUTUNG

if zahl % 2 == 0:
  Anweisung 1
else:
  Anweisung 2
	Prüft, ob der Wert der Variablen »zahl« durch 2 teilbar ist.
	In diesem Fall wird Anweisung 1 ausgeführt, sonst Anweisung 2.

Rote Karten:
IF-ANWEISUNGEN MIT LOGISCHEN OPERATOREN

	BEDEUTUNG
zahl > 0 and zahl <= 20 zahl < 0 or zahl > 10	Mithilfe der logischen Operatoren UND (and) und ODER (or) können Vergleiche kombiniert werden. Beim Operator UND müssen *beide* Vergleiche bzw. Aussagen wahr sein, damit dies auch auf die kombinierte Aussage zutrifft. Beim Operator ODER genügt es, wenn *mindestens* einer der beiden Vergleiche wahr ist.

Dunkelrote Karten:
FOR-SCHLEIFEN

	BEDEUTUNG

for i in range(Endwert):
  Anweisung
 	
Beispiel:
for i in range(3):
  Anweisung
	Übergibt man der Range-Funktion nur *einen* Parameter, so ist der Startwert 0 und der Endwert die als Endwert angegebene Zahl – 1.
	Im Beispiel wird die Anweisung genau 3-mal ausgeführt, nämlich für die Werte i=0, i=1 und i=2.

for i in range(Startwert, Endwert):
  Anweisung
 	
Beispiel:
for i in range(5,8):
  Anweisung
	Übergibt man der Range-Funktion *zwei* Parameter, so ist der Startwert die als Startwert angegebene Zahl und der Endwert die als Endwert angegebene Zahl – 1.
	Im Beispiel wird die Anweisung genau 3-mal ausgeführt, nämlich für die Werte i=5, i=6 und i=7.

for i in range(Startwert, Endwert, Schrittweite): Anweisung	Übergibt man der Range-Funktion *drei* Parameter, so sind die ersten beiden Start- und Endwert wie oben, während der dritte die Schrittweite ist. Die Schrittweite wird ohne Angabe auf 1 gesetzt und kann auch negativ sein. Bei negativer Schrittweite ist Endwert + 1 der letzte Wert, für den es ggf. einen Schleifendurchlauf geben kann (bei positiver Schrittweite ist es Endwert – 1).

Beispiel 1: for i in range(3, 0, –1): Anweisung	In Beispiel 1 wird die Anweisung genau 3-mal ausgeführt, nämlich für die Werte i=3, i=2 und i=1.
Beispiel 2: for i in range(4, 10, 2): Anweisung	In Beispiel 2 wird die Anweisung genau 3-mal ausgeführt, nämlich für die Werte i=4, i=6 und i=8.

Rosa Karten:

WHILE-SCHLEIFEN	BEDEUTUNG
while Bedingung: Anweisung(en)	Führt die folgende(n) durch Einrückung gekennzeichnete(n) Anweisung(en) aus, solange die Bedingung erfüllt ist.
Beispiel: i=0 while i*i < zahl: i = i + 1 zahl = zahl + i	Im Beispiel wird nur die Anweisung »i = i + 1« wiederholt ausgeführt. Die Anweisung »zahl = zahl + i« wird nach Beenden der Schleife *einmalig* ausgeführt.

Violette Karten: FOR-SCHLEIFEN MIT IF-ANWEISUNG IM SCHLEIFENRUMPF

	BEDEUTUNG

Beispiel:
x = 0
for i in range(30):
  if i % 5 == 0:
  x = x + 1
zahl = zahl + x
	Im Beispiel durchläuft »i« die Zahlen von 0 bis (einschließlich) 29. Nur bei den durch 5 teilbaren Zahlen wird »x« um 1 erhöht. Da 0, 5, 10, 15, 20 und 25 durch 5 teilbar sind, wird die Variable »zahl« insgesamt also um 6 erhöht.

Graue Karten: LISTEN

	BEDEUTUNG
a = [15,12,7,10,5,6,8] zahl = a[3]	Die Liste wird deklariert und mit sieben Zahlen initialisiert. Die Zeile »zahl = a[3]« weist der Variablen »zahl« den Wert des **vierten** Elements der Liste zu. Position | 1 | 2 | 3 | **4** | 5 | ... Index | 0 | 1 | 2 | **3** | 4 | ...
a = [0]*10 for i in range(len(a)): a[i] = i*i zahl = a[2]	Die Liste wird deklariert und mit 10 Nullen gefüllt. Die Schleife füllt die Liste mit den Quadraten der Indizes. Abschließend wird der Wert der Variablen »zahl« auf den Wert des **dritten** Elements der Liste gesetzt, also auf den Wert 4.

hopp-foundation.de