02 Operatoren, logische Ausdrücke, Struktogramme

01 Rechenoperationen

Um mit Zahlen rechnen zu können, stellt Python (wie auch andere Programmiersprachen in ähnlicher Art und Weise) einige Rechenoperatoren zur Verfügung:

Operator	Beschreibung	Beispiel	Ergebnis
+	Addition	5 + 3	8
-	Subtraktion	5 - 3	2
*	Multiplikation	5 * 3	15
/	Gleitkommadivision	5 / 3	1.6666666666666667
//	Ganzzahldivision	5 // 3	1
%	Modulo (Rest bei Division)	5 % 3	2
**	Potenzierung	5 ** 3	125

Hier ein Beispiel:

In [ ]:
```
a = 11
b = 3

# Ausgabe
print(f"Die Summe von {a} und {b} wird in Python mit {a}+{b} berechnet. Das Ergebnis ist {a+b}.")
print(f"Die Differenz von {a} und {b} wird in Python mit {a}-{b} berechnet. Das Ergebnis ist {a-b}.")
print(f"Das Produkt von {a} und {b} wird in Python mit {a}*{b} berechnet. Das Ergebnis ist {a*b}.")
print(f"Der Quotient von {a} und {b} wird in Python mit {a}/{b} berechnet. Das Ergebnis ist {a/b}.")
print(f"Der ganzzahlige Quotient von {a} und {b} wird in Python mit {a}//{b} berechnet. Das Ergebnis ist {a//b}.")
print(f"Der Rest der Ganzzahldivision von {a} und {b} wird in Python mit {a}%{b} berechnet. Das Ergebnis ist {a%b}.")
print(f"Die Potenz {a} hoch {b} wird in Python mit {a}**{b} berechnet. Das Ergebnis ist {a**b}.")
```

```
Die Summe von 11 und 3 wird in Python mit 11+3 berechnet. Das Ergebnis ist 14.
Die Differenz von 11 und 3 wird in Python mit 11-3 berechnet. Das Ergebnis ist 8.
Das Produkt von 11 und 3 wird in Python mit 11*3 berechnet. Das Ergebnis ist 33.
Der Quotient von 11 und 3 wird in Python mit 11/3 berechnet. Das Ergebnis ist 3.6666666666666665.
Der ganzzahlige Quotient von 11 und 3 wird in Python mit 11//3 berechnet. Das Ergebnis ist 3.
Der Rest der Ganzzahldivision von 11 und 3 wird in Python mit 11%3 berechnet. Das Ergebnis ist 2.
Die Potenz 11 hoch 3 wird in Python mit 11**3 berechnet. Das Ergebnis ist 1331.
```

Klammern

Auch in der Programmierung gibt es eine "Hierarchie der Rechenoperatoren. Beispielsweise gilt "Punkt vor Strich". Daher kann man mit runden Klammern () Rechenausdrücke anpassen.

In [ ]:
```
a = 5
b = 6
c = 7

print(f"{a} + {b} * {c} = {a} + {b*c} = {a+b*c}")
print(f"({a} + {b}) * {c} = {a+b} * {c} = {(a+b)*c}")
```

```
5 + 6 * 7 = 5 + 42 = 47
(5 + 6) * 7 = 11 * 7 = 77
```

Aufgaben

1. Der FC Sparsheim möchte ein neues Fußballfeld anlegen und berechnen, wie teuer die Anschaffung wird. Ein Quadratmeter Rasen kostet 20€. Ein Tor kostet 1400€. Mehr Kosten sollen zunächst nicht berücksichtigt werden. Schreibe ein Programm, das nach der Länge und Breite des Feldes fragt und die Gesamtkosten für das Fußballfeld nennt.

In [ ]:
```
"""Aufgabe a"""
```

2. Auf einer Party wurden Pizzen bestellt. Da so viele Leute da sind, werden einfach reihum Pizzastücke ausgeteilt, bis für eine neue Austeilrunde nicht mehr genügend Pizzastücke übrig sind. Erstelle ein Programm, das zu Beginn die Anzahl der Personen und der Pizzastücke erfragt und berechnet, wie viele Pizzastücke jede Person erhält und wie viele Pizzastücke am Ende noch übrig bleiben.

In [ ]:
```
"""Aufgabe b"""
```

3. *Für Schnelle*: In dieser Aufgabe wurde ein Timer erstellt, der in 1-Sekunden-Schritten immer eine Zahl hochzählt. Der Timer beginnt bei einer Zufallszahl. Ergänze das Programm so, dass damit die Uhrzeit angezeigt wird. Die Zahl 0 der Zahl bedeutet dabei 0:00 Uhr. (Für besondere Ästheten: ein String str lässt sich mit str.zfill(2) mit führenden 0en schreiben.)

In [ ]:
```
"""Aufgabe c"""

import time
import random
from IPython.display import clear_output

# Anzahl Sekunden des Tags
TAG_SEKUNDEN = 86400  # z.B. 24 Stunden = 86400 Sekunden

# Zufallszahl zwischen 0 und Anzahl Sekunden des Tags
zufallszahl = random.randint(0, TAG_SEKUNDEN)

# Schleife, die jede Sekunde läuft
while True:
    # Ausgabe in der Zelle löschen
    clear_output(wait=True)

    # Ausgabe der Zufallszahl zur Kontrolle
    print(f"zufallszahl = {zufallszahl}")

    """ Hier kommt dein Code hin"""

    # Zufallszahl erhöhen
    zufallszahl += 1
    # 1 Sekunde pausieren
    time.sleep(1)
```

02 Operatoren, logische Ausdrücke, Struktogramme

01 Rechenoperationen (Lösung)

Um mit Zahlen rechnen zu können, stellt Python (wie auch andere Programmiersprachen in ähnlicher Art und Weise) einige Rechenoperatoren zur Verfügung:

Operator	Beschreibung	Beispiel	Ergebnis
+	Addition	5 + 3	8
-	Subtraktion	5 - 3	2
*	Multiplikation	5 * 3	15
/	Gleitkommadivision	5 / 3	1.6666666666666667
//	Ganzzahldivision	5 // 3	1
%	Modulo (Rest bei Division)	5 % 3	2
**	Potenzierung	5 ** 3	125

Hier ein Beispiel:

In [ ]:
```
a = 11
b = 3

# Ausgabe
print(f"Die Summe von {a} und {b} wird in Python mit {a}+{b} berechnet. Das Ergebnis ist {a+b}.")
print(f"Die Differenz von {a} und {b} wird in Python mit {a}-{b} berechnet. Das Ergebnis ist {a-b}.")
print(f"Das Produkt von {a} und {b} wird in Python mit {a}*{b} berechnet. Das Ergebnis ist {a*b}.")
print(f"Der Quotient von {a} und {b} wird in Python mit {a}/{b} berechnet. Das Ergebnis ist {a/b}.")
print(f"Der ganzzahlige Quotient von {a} und {b} wird in Python mit {a}//{b} berechnet. Das Ergebnis ist {a//b}.")
print(f"Der Rest der Ganzzahldivision von {a} und {b} wird in Python mit {a}%{b} berechnet. Das Ergebnis ist {a%b}.")
print(f"Die Potenz {a} hoch {b} wird in Python mit {a}**{b} berechnet. Das Ergebnis ist {a**b}.")
```

```
Die Summe von 11 und 3 wird in Python mit 11+3 berechnet. Das Ergebnis ist 14.
Die Differenz von 11 und 3 wird in Python mit 11-3 berechnet. Das Ergebnis ist 8.
Das Produkt von 11 und 3 wird in Python mit 11*3 berechnet. Das Ergebnis ist 33.
Der Quotient von 11 und 3 wird in Python mit 11/3 berechnet. Das Ergebnis ist 3.6666666666666665.
Der ganzzahlige Quotient von 11 und 3 wird in Python mit 11//3 berechnet. Das Ergebnis ist 3.
Der Rest der Ganzzahldivision von 11 und 3 wird in Python mit 11%3 berechnet. Das Ergebnis ist 2.
Die Potenz 11 hoch 3 wird in Python mit 11**3 berechnet. Das Ergebnis ist 1331.
```

Klammern

Auch in der Programmierung gibt es eine "Hierarchie der Rechenoperatoren. Beispielsweise gilt "Punkt vor Strich". Daher kann man mit runden Klammern () Rechenausdrücke anpassen.

In [ ]:
```
a = 5
b = 6
c = 7

print(f"{a}+{b}*{c} = {a} + {b*c} = {a+b*c}")
print(f"({a}+{b})*{c} = {a+b} * {c} = {(a+b)*c}")
```

```
5+6*7 = 5 + 42 = 47
(5+6)*7 = 11 * 7 = 77
```

Aufgaben

1. Der FC Sparsheim möchte ein neues Fußballfeld anlegen und berechnen, wie teuer die Anschaffung wird. Ein Quadratmeter Rasen kostet 20€. Ein Tor kostet 1400€. Mehr Kosten sollen zunächst nicht berücksichtigt werden. Schreibe ein Programm, das nach der Länge und Breite des Feldes fragt und die Gesamtkosten für das Fußballfeld nennt.

In [ ]:
```
"""Aufgabe a"""

# Eingaben zur Feldgröße
laenge = int(input("Länge des Fußballfelds in Metern: "))
breite = int(input("Breite des Fußballfelds in Metern: "))

# Berechnung der Fläche in qm
flaeche = laenge * breite

# Berechnung der Kosten inkl. Kosten für 2 Tore
kosten = flaeche * 20 + 2 * 1400

# Ausgabe
print(f"Ein Feld, das {laenge}m lang und {breite}m breit ist kostet inklusive der beiden Tore {kosten} Euro.")
```

```
Ein Feld, das 100m lang und 50m breit ist kostet inklusive der beiden Tore 102800 Euro.
```

2. Auf einer Party wurden Pizzen bestellt. Da so viele Leute da sind, werden einfach reihum Pizzastücke ausgeteilt, bis für eine neue Austeilrunde nicht mehr genügend Pizzastücke übrig sind. Erstelle ein Programm, das zu Beginn die Anzahl der Personen und der Pizzastücke erfragt und berechnet, wie viele Pizzastücke jede Person erhält und wie viele Pizzastücke am Ende noch übrig bleiben.

In [ ]:
```
"""Aufgabe b"""

# Eingaben
anzahl_personen = int(input("Wie viele Leute sind auf der Party? "))
anzahl_pizzastuecke = int(input("Wie viele Pizzastücke wurden bestellt? "))

# Berechnung der Stücke pro Person
# Hier hilft die Ganzzahldivision, die nur ganze Stücke berechnet.
stuecke_pro_person = anzahl_pizzastuecke // anzahl_personen

# Berechnung der übrigen Stücke
# Mit dem Modulo-Operator wird der Rest der Pizzastücke leicht berechnet.
uebrige_pizzastuecke = anzahl_pizzastuecke % anzahl_personen

# Ausgabe
print(f"Auf der Party sind {anzahl_personen} Personen.")
print(f"Es wurden {anzahl_pizzastuecke} Stücke Pizza bestellt.")
print(f"Jede Person erhält {stuecke_pro_person} Stücke Pizza.")
print(f"Nach dem Verteilen der Stücke bleiben {uebrige_pizzastuecke} Stücke Pizza übrig.")
```

```
Auf der Party sind 15 Personen.
Es wurden 87 Stücke Pizza bestellt.
Jede Person erhält 5 Stücke Pizza.
Nach dem Verteilen der Stücke bleiben 12 Stücke Pizza übrig.
```

3. *Für Schnelle*: In dieser Aufgabe wurde ein Timer erstellt, der in 1-Sekunden-Schritten immer eine Zahl hochzählt. Der Timer beginnt bei einer Zufallszahl. Ergänze das Programm so, dass damit die Uhrzeit angezeigt wird. Die Zahl 0 der Zahl bedeutet dabei 0:00 Uhr. (Für besondere Ästheten: ein String str lässt sich mit str.zfill(2) mit führenden 0en schreiben.)

In [ ]:
```
"""Aufgabe c"""

import time
import random
from IPython.display import clear_output

# Anzahl Sekunden des Tags
TAG_SEKUNDEN = 86400  # z.B. 24 Stunden = 86400 Sekunden

# Zufallszahl zwischen 0 und Anzahl Sekunden des Tags
zufallszahl = random.randint(0, TAG_SEKUNDEN)

# Schleife, die jede Sekunde läuft
while True:
    # Ausgabe in der Zelle löschen
    clear_output(wait=True)

    # Ausgabe der Zufallszahl zur Kontrolle
    print(f"zufallszahl = {zufallszahl}")

    """ Hier kommt dein Code hin"""
    # Berechnung der Sekunden, Minuten und Stunden
    sekunden = zufallszahl % 60
    minuten = (zufallszahl // 60) % 60
    stunden = (zufallszahl // 3600) % 24

    # Formattierung für schöne Ausgabe
    sekunden = str(sekunden).zfill(2)
    minuten = str(minuten).zfill(2)
    stunden = str(stunden).zfill(2)

    print(f"Timer: {stunden}:{minuten}:{sekunden}")

    # Zufallszahl erhöhen
    zufallszahl += 1
    # 1 Sekunde pausieren
    time.sleep(1)
```

02 Operatoren, logische Ausdrücke, Struktogramme

02 Logische Ausdrücke

Zur Erinnerung: Ein Boolean-Wert ist ein Wahrheitswert, der entweder wahr (`True`) oder falsch (`False`) ist. Zum Beispiel:

In [ ]:
```
regen_gemeldet = False
sonnig = True
```

Entsprechend können Zahlen auch miteinander verglichen werden mit den folgenden *Vergleichsoperatoren*.

Operator	Beschreibung	Beispiel	Ergebnis
<	kleiner als	5 < 3	False
>	größer als	5 > 3	True
<=	kleiner gleich	5 <= 3	False
>=	größer gleich	5 >= 3	True
!=	ungleich	5 != 3	True
==	gleich	5 == 3	False

Probiere es selbst aus, indem du die Zahlenwerte für a und b änderst.

In [ ]:
```
a = 11
b = 3

# Ausgabe
print(f"Ist {a} kleiner als {b}? Das kann man mit {a}<{b} herausfinden. Antwort: {a<b}.")
print(f"Ist {a} größer als {b}? Das kann man mit {a}>{b} herausfinden. Antwort: {a>b}.")
print(f"Ist {a} kleiner gleich {b}? Das kann man mit {a}<={b} herausfinden. Antwort: {a<=b}.")
print(f"Ist {a} größer gleich {b}? Das kann man mit {a}>={b} herausfinden. Antwort: {a>=b}.")
print(f"Ist {a} nicht gleich {b}? Das kann man mit {a}!={b} herausfinden. Antwort: {a!=b}.")
print(f"Ist {a} gleich {b}? Das kann man mit {a}=={b} herausfinden. Antwort: {a==b}.")
```

```
Ist 11 kleiner als 3? Das kann man mit 11<3 herausfinden. Antwort: False.
Ist 11 größer als 3? Das kann man mit 11>3 herausfinden. Antwort: True.
Ist 11 kleiner gleich 3? Das kann man mit 11<=3 herausfinden. Antwort: False.
Ist 11 größer gleich 3? Das kann man mit 11>=3 herausfinden. Antwort: True.
Ist 11 nicht gleich 3? Das kann man mit 11!=3 herausfinden. Antwort: True.
Ist 11 gleich 3? Das kann man mit 11==3 herausfinden. Antwort: False.
```

Vorsicht: Einer der häufigsten Fehlerquellen in der Programmierung ist, dass der Zuweisungsoperator $=$ mit dem Gleichheitsoperator $==$ verwechselt wird!!!

Manchmal ist man auch daran interessiert, ob eine Aussage nicht wahr ist, ob zwei Aussagen wahr sind oder ob mindestens eine von zwei Aussagen wahr ist. Dafür gibt es die *logischen Operatoren* `not` , `and` und `or` .

In [ ]:
```
regen = True
sonne = False

kein_regen = not regen
regenbogen = (regen and sonne)
regen_oder_sonne = (regen or sonne)

print(f"kein_regen = {kein_regen}")
print(f"regenbogen = {regenbogen}")
print(f"regen_oder_sonne = {regen_oder_sonne}")
```

```
kein_regen = False
regenbogen = False
regen_oder_sonne = True
```

Aufgaben

1. Schreibe ein Programm, das zwei Zahlen als Eingabe erhält und True ausgibt, wenn die erste Zahl größer ist und sonst False.

In [ ]:
```
"""Aufgabe a."""
```

2. Schreibe ein Programm, das zwei Zahlen erhält und True ausgibt, wenn die erste Zahl durch die zweite Zahl teilbar ist und sonst False (Tipp: $\%$)

In [ ]:
```
"""Aufgabe b."""
```

3. *(Für Schnelle)* In dieser Aufgabe geht es um das Würfelspiel „Mäxle“. Dabei werden zwei Würfel gleichzeitig geworfen und jede Spielerin / jeder Spieler hat das Ziel, einen „höheren“ Wurf als die Vorgängerin bzw. der Vorgänger zu werfen.
   Bei einem Wurf ist stets die höhere Augenzahl die Zehnerstelle und die niedrigere die Einerstelle. Die beiden Augenzahlen 3 und 5 ergeben also z.B. die Zahl 53.
   Zwei gleiche Augenzahlen gelten als Pasch, die 1 und die 2 (also 21) als „Mäxle“. Die Reihenfolge ist: „normale Zahlen“ wie 31, 32, 41,..., 65; dann die Pässche 11, 22, ..., 66 und schließlich als höchstes Ergebnis das „Mäxle“ (21).
   Im folgenden Code wird ein Würfelwurf simuliert. Gib mit logischen Verknüpfungen an, ob folgende Ereignisse wahr oder falsch sind:

   A. Beide Würfel zeigen die Augenzahl 4.
   B. Es wird ein Pasch oder das „Mäxle“ gewürfelt.
   C. Es wird eine „normale Zahl“, aber nicht das „Mäxle“ gewürfelt.
   D. Es wird eine „normale Zahl“ über 60 gewürfelt. (D.h. der Sechserpasch ist nicht dabei)
   E. Denke dir noch weitere Ereignisse aus und formuliere dazu die passenden Aussagen (als Kommentare im Code)

In [ ]:
```
"""Aufgabe für Schnelle"""
import random

# Zufallszahl zwischen 1 und 6 für beide Würfel
wuerfel_1 = random.randint(1, 6)
wuerfel_2 = random.randint(1, 6)

# Ausgabe des Würfelwurfs
print(f"Gewürfelt wurde {wuerfel_1}{wuerfel_2}")

# Beide Würfel zeigen die Augenzahl 4.

# Es wird ein Pasch oder das „Mäxle“ gewürfelt.

# Es wird eine „normale Zahl“, aber nicht das „Mäxle“ gewürfelt.

# Es wird eine „normale Zahl“ über 60 gewürfelt. (D.h. der Sechserpasch ist nicht dabei)
```

02 Operatoren, logische Ausdrücke, Struktogramme

02 Logische Ausdrücke

Zur Erinnerung: Ein Boolean-Wert ist ein Wahrheitswert, der entweder wahr (`True`) oder falsch (`False`) ist. Zum Beispiel:

In [ ]:
```
regen_gemeldet = False
sonnig = True
```

Entsprechend können Zahlen auch miteinander verglichen werden mit den folgenden *Vergleichsoperatoren*.

Operator	Beschreibung	Beispiel	Ergebnis
<	kleiner als	5 < 3	False
>	größer als	5 > 3	True
<=	kleiner gleich	5 <= 3	False
>=	größer gleich	5 >= 3	True
!=	ungleich	5 != 3	True
==	gleich	5 == 3	False

Probiere es selbst aus, indem du die Zahlenwerte für a und b änderst.

In [ ]:
```
a = 11
b = 3

# Ausgabe
print(f"Ist {a} kleiner als {b}? Das kann man mit {a}<{b} herausfinden. Antwort: {a<b}.")
print(f"Ist {a} größer als {b}? Das kann man mit {a}>{b} herausfinden. Antwort: {a>b}.")
print(f"Ist {a} kleiner gleich {b}? Das kann man mit {a}<={b} herausfinden. Antwort: {a<=b}.")
print(f"Ist {a} größer gleich {b}? Das kann man mit {a}>={b} herausfinden. Antwort: {a>=b}.")
print(f"Ist {a} nicht gleich {b}? Das kann man mit {a}!={b} herausfinden. Antwort: {a!=b}.")
print(f"Ist {a} gleich {b}? Das kann man mit {a}=={b} herausfinden. Antwort: {a==b}.")
```

```
Ist 11 kleiner als 3? Das kann man mit 11<3 herausfinden. Antwort: False.
Ist 11 größer als 3? Das kann man mit 11>3 herausfinden. Antwort: True.
Ist 11 kleiner gleich 3? Das kann man mit 11<=3 herausfinden. Antwort: False.
Ist 11 größer gleich 3? Das kann man mit 11>=3 herausfinden. Antwort: True.
Ist 11 nicht gleich 3? Das kann man mit 11!=3 herausfinden. Antwort: True.
Ist 11 gleich 3? Das kann man mit 11==3 herausfinden. Antwort: False.
```

Vorsicht: Einer der häufigsten Fehlerquellen in der Programmierung ist, dass der Zuweisungsoperator $=$ mit dem Gleichheitsoperator $==$ verwechselt wird!!!

Manchmal ist man auch daran interessiert, ob eine Aussage nicht wahr ist, ob zwei Aussagen wahr sind oder ob mindestens eine von zwei Aussagen wahr ist. Dafür gibt es die *logischen Operatoren* `not` , `and` und `or` .

In [ ]:
```
regen = True
sonne = False

kein_regen = not regen
regenbogen = (regen and sonne)
regen_oder_sonne = (regen or sonne)

print(f"kein_regen = {kein_regen}")
print(f"regenbogen = {regenbogen}")
print(f"regen_oder_sonne = {regen_oder_sonne}")
```

```
kein_regen = False
regenbogen = False
regen_oder_sonne = True
```

Aufgaben

1. Schreibe ein Programm, das zwei Zahlen als Eingabe erhält und True ausgibt, wenn die erste Zahl größer ist und sonst False.

In [ ]:
```
"""Aufgabe a."""

# Eingaben
zahl_1 = float(input("Wie lautet die erste Zahl? "))
zahl_2 = float(input("Wie lautet die zweite Zahl? "))

# Ausgabe
print(f"{zahl_1} > {zahl_2} ergibt den Wahrheitswert {zahl_1 > zahl_2}.")
```

```
4.0 > 5.0 ergibt den Wahrheitswert False.
```

2. Schreibe ein Programm, das zwei Zahlen erhält und True ausgibt, wenn die erste Zahl durch die zweite Zahl teilbar ist und sonst False (Tipp: $\%$)

In [ ]:
```
"""Aufgabe b."""

# Eingaben
zahl_1 = int(input("Wie lautet die erste Zahl? "))
zahl_2 = int(input("Wie lautet die zweite Zahl? "))

# Ergebnis in Variable speichern
# Der Modulo-Operator % ist genau dann 0, wenn kein Rest bei der Division entsteht.
ist_teilbar = ((zahl_1 % zahl_2) == 0)

# Ausgabe
print(f"Wahrheitswert, ob {zahl_1} durch {zahl_2} teilbar ist: {ist_teilbar}")
```

```
Wahrheitswert, ob 18 durch 3 teilbar ist: True
```

3. *(Für Schnelle)* In dieser Aufgabe geht es um das Würfelspiel „Mäxle“. Dabei werden zwei Würfel gleichzeitig geworfen und jede Spielerin / jeder Spieler hat das Ziel, einen „höheren“ Wurf als die Vorgängerin bzw. der Vorgänger zu werfen.
   Bei einem Wurf ist stets die höhere Augenzahl die Zehnerstelle und die niedrigere die Einerstelle. Die beiden Augenzahlen 3 und 5 ergeben also z.B. die Zahl 53.
   Zwei gleiche Augenzahlen gelten als Pasch, die 1 und die 2 (also 21) als „Mäxle“. Die Reihenfolge ist: „normale Zahlen“ wie 31, 32, 41,..., 65; dann die Pärsche 11, 22, ..., 66 und schließlich als höchstes Ergebnis das „Mäxle“ (21).
   Im folgenden Code wird ein Würfelwurf simuliert. Gib mit logischen Verknüpfungen an, ob folgende Ereignisse wahr oder falsch sind:

   A. Beide Würfel zeigen die Augenzahl 4.
   B. Es wird ein Pasch oder das „Mäxle“ gewürfelt.
   C. Es wird eine „normale Zahl“, aber nicht das „Mäxle“ gewürfelt.
   D. Es wird eine „normale Zahl“ über 60 gewürfelt. (D.h. der Sechserpasch ist nicht dabei)
   E. Denke dir noch weitere Ereignisse aus und formuliere dazu die passenden Aussagen (als Kommentare im Code)

In [ ]:
```
"""Aufgabe für Schnelle"""
import random

# Zufallszahl zwischen 1 und 6 für beide Würfel
wuerfel_1 = random.randint(1, 6)
wuerfel_2 = random.randint(1, 6)

# Ausgabe des Würfelwurfs
print(f"Gewürfelt wurde {wuerfel_1}{wuerfel_2}")

# Beide Würfel zeigen die Augenzahl 4.
beide_4 = ((wuerfel_1 == 4) and (wuerfel_2 == 4))
print(f"Beide Wuerfel 4? --> {beide_4}")

# Es wird ein Pasch oder das „Mäxle“ gewürfelt.
pasch_oder_maexle = ((wuerfel_1 == wuerfel_2) or (wuerfel_1 == 1 and wuerfel_2 == 2) or (wuerfel_1 == 2 and wuerfel_2 == 1))
print(f"Pasch oder Mäxle? --> {pasch_oder_maexle}")

# Es wird eine „normale Zahl“, aber nicht das „Mäxle“ gewürfelt.
normal_nicht_maexle = (not pasch_oder_maexle)
print(f"Normale Zahl und nicht Mäxle? --> {normal_nicht_maexle}")

# Es wird eine „normale Zahl“ über 60 gewürfelt. (D.h. der Sechserpasch ist nicht dabei)
normal_ueber_60 = ((wuerfel_1 == 6 and wuerfel_2 != 6) or (wuerfel_1 != 6 and wuerfel_2 == 6))
print(f"Normale Zahl über 60? --> {normal_ueber_60}")
```

```
Gewürfelt wurde 21
Beide Wuerfel 4? --> False
Pasch oder Mäxle? --> True
Normale Zahl und nicht Mäxle? --> False
Normale Zahl über 60? --> False
```

02 Operatoren, logische Ausdrücke, Struktogramme

03 Struktogramme

Die Programme, welche du später schreiben wirst, werden immer komplizierter. Man kann dabei schnell die Übersicht über den eigentlichen Algorithmus verlieren. Damit du dich voll auf die Erstellung konzentrieren und auch die Strukturen besser erkennen kannst, nutzen wir *Struktogramme*. Diese sind eine blockartige Darstellung von Algorithmen. Der Struktogrammeditor ist ein Tool des Lehrstuhls für Didaktik der Informatik der TU Dresden (DDI), mit dem solche Struktogramme erstellt werden können.

Aufgaben

Mache dich eigenständig mit dem Tool zur Erstellung von Struktogrammen auf der Seite der TU Dresden vertraut und erstelle für mindestens zwei deiner hier erstellten Programme ein Struktogramm. Speichere das Struktogramm anschließend im Informatik-Ordner ab.

Du benötigst zunächst nur das *Eingabe-Feld*, das *Ausgabe-Feld* und die *Anweisung*.

Seite der TU Dresden

02 Operatoren, logische Ausdrücke, Struktogramme

04 Übung

Arbeitsauftrag: Sammle mindestens 5 ★ durch die Bearbeitung der folgenden Aufgaben.

Aufgabe 1 (★)

Gib für jede Kombnation aus `True` und `False` von `a` und `b` den Wahrheitswert des folgenden Ausdrucks an.

```
(a and b) or ((not b) and a)
```

Lösung:

a = True; b = True -->

a = True; b = False -->

a = False; b = True -->

a = False; b = False -->

Aufgabe 2 (★)

Eine Bäckerei verkauft Brötchen für 75 Cent, Puddingteilchen für 1,40 Euro und Brote für 3,40 Euro. Implementiere ein Programm, das je nach eingekaufter Menge der drei Produkte die Gesamtrechnung berechnet.

In [ ]:

Aufgabe 3 (★)

Implementiere ein Programm, das die beiden Längen der beiden Katheten eines rechtwinkligen Dreiecks erhält und die Länge der Hypothenuse berechnet.

▶ Tipp 1
▶ Tipp 2
▶ Tipp 3

In [ ]:

Aufgabe 4 (★ ★)

Implemetiere ein Programm, das die Ecken eines regelmäßigen Polygons (Dreieck, Viereck, Fünfeck, ...) als Eingabe erhält, und die Summe aller Innenwinkel des Polygons berechnet.

▶ Tipp

In [ ]:

Aufgabe 5 (★ ★)

Ein Hund ist ein Tier, das entweder knurrt oder hechelt und stets knuddelig aussieht. Implementiere ein Programm, das danach fragt, ob eine Sache ein Tier ist, knurrt, hechelt und/oder knuddelig aussieht (man soll 0 eingeben, wenn nein und 1 wenn ja) und True ausgibt, wenn es sich dabei um einen Hund handelt.

In [ ]:

Aufgabe 6 (★ ★)

Der Wurf eines Basketballs kann mit der quadratischen Funktion $f(x) = -0,12x^2 + x + 2$ modelliert werden, wobei x die horizontale Entfernung vom Abwurf und $f(x)$ die Höhe des Balls in Metern darstellen.

Implementiere ein Programm, dass die horizontale Entfernung des Balls als Eingabe erhält und damit die Höhe des Balls bestimmt. Gib außerdem aus, ob der Ball an der gegebenen Position sein kann oder nicht.

In [ ]:

Aufgabe 7 (★ ★ ★)

Die de-Morgan'schen Regeln sind wichtige Regeln in der Logik. Sie besagen folgendes:

1. Das Gegenteil davon, dass zwei Aussagen gleichzeitig wahr sind, ist identisch mit der Aussage, dass das Gegenteil der einen oder das Gegenteil der anderen Aussage wahr sind (oder das Gegenteil beider Aussagen).
2. Das Gegenteil davon, dass eine Aussage oder eine andere Aussage (oder beide) wahr sind, ist identisch mit der Aussage, dass das Gegenteil der einen und das Gegenteil der anderen Aussage wahr sind.

Stelle die de-Morgan'schen Regeln in Python dar und prüfe sie für alle 4 Kombinationen von Wahrheitswerten für die beiden benötigten Aussagen.

▶ Tipp 1
▶ Tipp 2

In [ ]:

```
aussage_1 = True
aussage_2 = True

# 1. de-Morgan'sche Regel

# 2. de-Morgan'sche Regel
```

02 Operatoren, logische Ausdrücke, Struktogramme

04 Übung (Lösung)

Arbeitsauftrag: Sammle mindestens 5 ★ durch die Bearbeitung der folgenden Aufgaben.

Aufgabe 1 (★)

Gib für jede Kombnation aus `True` und `False` von `a` und `b` den Wahrheitswert des folgenden Ausdrucks an.

```
(a and b) or ((not b) and a)
```

Lösung:

a = True; b = True --> `(True and True) or ((not True) and True) == True or (False and True) == True or False == True`

a = True; b = False --> `(True and False) or ((not False) and True) == False or (True and True) == False or True == True`

a = False; b = True --> `(False and True) or ((not True) and False) == False or (True and False) == False or False == False`

a = False; b = False --> `(False and False) or ((not False) and False) == False or (True and False) == False or False == False`

Aufgabe 2 (★)

Eine Bäckerei verkauft Brötchen für 75 Cent, Puddingteilchen für 1,40 Euro und Brote für 3,40 Euro. Implementiere ein Programm, das je nach eingekaufter Menge der drei Produkte die Gesamtrechnung berechnet.

In [ ]:
```
# Eingaben
anz_broetchen = int(input("Wie viele Brötchen wurden verkauft? "))
anz_puddingteilchen = int(input("Wie viele Puddingteilchen wurden verkauft? "))
anz_brot = int(input("Wie viele Brote wurden verkauft? "))

# Preise werden extra abgespeichert, so könnten sie bei Preisänderungen schnell geändert werden
preis_broetchen = 0.75
preis_puddingteilchen = 1.40
preis_brot = 3.40

# Berechnung des Gesamtpreises
gesamtpreis = anz_broetchen * preis_broetchen + anz_puddingteilchen * preis_puddingteilchen + anz_brot * preis_brot

# Ausgabe
print(f"Die {anz_broetchen} Brötchen, {anz_puddingteilchen} Puddingteilchen und {anz_brot} Brote kosten zusammen {gesamtpreis} Euro.")
```

```
Die 5 Brötchen, 2 Puddingteilchen und 1 Brote kosten zusammen 9.95 Euro.
```

Aufgabe 3 (★)

Implementiere ein Programm, das die beiden Längen der beiden Katheten eines rechtwinkligen Dreiecks erhält und die Länge der Hypothenuse berechnet.

▶ Tipp 1
▶ Tipp 2
▶ Tipp 3

In [ ]:
```
# Länge der beiden Kathen
len_kathete_1 = float(input("Wie lang ist die erste Kathete? "))
len_kathete_2 = float(input("Wie lang ist die zweite Kathete? "))

# Berechnung der Länge der Hypothenuse
len_hypothenuse = (len_kathete_1**2 + len_kathete_2**2)**(1/2)

# Ausgabe
print(f"Die Hypothenuse eines rechtwinkligen Dreiecks, dessen Katheten die Längen {len_kathete_1} und {len_kathete_2} haben, hat die Länge {len_hyp
```

```
Die Hypothenuse eines rechtwinkligen Dreiecks, dessen Katheten die Längen 4.0 und 5.0 haben, hat die Länge 6.4031242374328485.
```

Aufgabe 4 (★ ★)

Implemetiere ein Programm, das die Ecken eines regelmäßigen Polygons (Dreieck, Viereck, Fünfeck, ...) als Eingabe erhält, und die Summe aller Innenwinkel des Polygons berechnet.

▶ Tipp

In [ ]:
```
# Eingabe
anz_ecken = int(input("Wie viele Seiten hat das Polygon? "))

# Berechnung der Summe der Innenwinkel
sum_innenwinkel = (anz_ecken - 2) * 180

# Ausgabe
print(f"Die Summe aller Innenwinkel eines {anz_ecken}-Ecks beträgt {sum_innenwinkel} Grad.")
```

```
Die Summe aller Innenwinkel eines 7-Ecks beträgt 900 Grad.
```

Aufgabe 5 (★ ★)

Ein Hund ist ein Tier, das entweder knurrt oder hechelt und stets knuddelig aussieht. Implementiere ein Programm, das danach fragt, ob eine Sache ein Tier ist, knurrt, hechelt und/oder knuddelig aussieht (man soll 0 eingeben, wenn nein und 1 wenn ja) und True ausgibt, wenn es sich dabei um einen Hund handelt.

In [ ]:
```
# Eingaben
tier = bool(int(input("Ist das Ding ein Tier? (0 für nein, 1 für ja) ")))
knurrt = bool(int(input("Knurrt das Ding? (0 für nein, 1 für ja) ")))
hechelt = bool(int(input("Hechelt das Ding? (0 für nein, 1 für ja) ")))
knuddelig = bool(int(input("Sieht das Ding zum Knuddeln aus? (0 für nein, 1 für ja) ")))

# Berechnung, ob das Ding ein Hund ist
ist_hund = tier and ((knurrt and not hechelt) or (hechelt and not knurrt)) and knuddelig

# Ausgabe
print(f"Tier: {tier}")
print(f"Knurrt: {knurrt}")
print(f"Hechelt: {hechelt}")
print(f"Knuddelig: {knuddelig}")
print()
print(f"Hund: {ist_hund}")
```

```
Tier: True
Knurrt: True
Hechelt: False
Knuddelig: True

Hund: True
```

Aufgabe 6 (★ ★)

Der Wurf eines Basketballs kann mit der quadratischen Funktion $f(x) = -0,12x^2 + x + 2$ modelliert werden, wobei x die horizontale Entfernung vom Abwurf und $f(x)$ die Höhe des Balls in Metern darstellen.

Implementiere ein Programm, dass die horizontale Entfernung des Balls als Eingabe erhält und damit die Höhe des Balls bestimmt. Gib außerdem aus, ob der Ball an der gegebenen Position sein kann oder nicht.

In [ ]:
```
# Eingabe der horizontalen Entfernung
hor_entf = float(input("Welche horizontale Entfernung zum Abwurf hat der Ball? "))

# Berechnung der Höhe
hoehe = -0.12 * hor_entf**2 + hor_entf + 2

# Der Ball ist an einer legitimen Position, wenn er eine positive horizontale Enfernung hat und über der Erde ist
pos_legit = (hor_entf >= 0) and (hoehe >= 0)

# Ausgabe
print(f"Der Ball hat bei einer horizontalen Entfernung von {hor_entf} Metern eine Höhe von {hoehe} Metern.")
print(f"Die Position des Balls ist legitim: {pos_legit}.")
```

```
Der Ball hat bei einer horizontalen Entfernung von 4.0 Metern eine Höhe von 4.08 Metern.
Die Position des Balls ist legitim: True.
```

Aufgabe 7 (★ ★ ★)

Die de-Morgan'schen Regeln sind wichtige Regeln in der Logik. Sie besagen folgendes:

1. Das Gegenteil davon, dass zwei Aussagen gleichzeitig wahr sind, ist identisch mit der Aussage, dass das Gegenteil der einen oder das Gegenteil der anderen Aussage wahr sind (oder das Gegenteil beider Aussagen).
2. Das Gegenteil davon, dass eine Aussage oder eine andere Aussage (oder beide) wahr sind, ist identisch mit der Aussage, dass das Gegenteil der einen und das Gegenteil der anderen Aussage wahr sind.

Stelle die de-Morgan'schen Regeln in Python dar und prüfe sie für alle 4 Kombinationen von Wahrheitswerten für die beiden benötigten Aussagen.

▶ Tipp 1
▶ Tipp 2

In [ ]:
```
aussage_1 = False
aussage_2 = False

# 1. de-Morgan'sche Regel
de_morgan_1 = (not (aussage_1 and aussage_2)) == ((not aussage_1) or (not aussage_2))

# 2. de-Morgan'sche Regel
de_morgan_2 = (not (aussage_1 or aussage_2)) == ((not aussage_1) and (not aussage_2))

# Ausgabe (diese sollte immer True ausgeben, dann ist die Regel richtig)
print(f"Die 1. de-Morgan'sche Regeln ist {de_morgan_1}.")
print(f"Die 2. de-Morgan'sche Regeln ist {de_morgan_2}.")
```

```
Die 1. de-Morgan'sche Regeln ist True.
Die 2. de-Morgan'sche Regeln ist True.
```