Control Flow mit Python

Wir wollen mehr Kontrolle haben!

1.

Einführung

Was bedeutet Control Flow überhaupt

Passwort-Überprüfung

Es folgt ein kurzes Beispiel, welches wiederholte Eingaben verwendet und Bedingungen überprüft

Boolescher Datentyp

- Datentyp mit nur zwei möglichen Werten: **`Wahr und Falsch`**

```
light_is_on = True
door_is_open = False
```

Vergleichsoperatoren

Ausdruck	Beschreibung
x == y	Wird als Wahr ausgewertet, wenn x gleich y ist; sonst Falsch
x != y	Wird als Wahr ausgewertet, wenn x nicht dasselbe wie y ist; sonst Falsch
x < y	Wird als Wahr ausgewertet, wenn x kleiner als y ist; sonst Falsch
x > y	...
x >= y	...
x <= y	...

Logische Operatoren

Logical And: and	
Expression	**Value**
`True and True`	`True`
`True and False`	`False`
`False and True`	`False`
`False and False`	`False`

Logische Operatoren

Logical Or: `or`	
Expression	**Value**
`True or True`	`True`
`True or False`	`True`
`False or True`	`True`
`False or False`	`False`

Logische Operatoren

Logical Negation: `not`	
Expression	**Value**
`not True`	`False`
`not False`	`True`

Aktualisierte Operatorenreihenfolge

Reihenfolge	Operator	Operation	Beispiel
5	*	Multiplikation	3 * 4
6	-	Subtraktion	10 - 8
7	+	Addition	100 + 200
8	<, <=, >, >=, !=, ==	Vergleich	3 > 2
9	not	Negation	x is not y
10	and	Logical And	x == 2 and y == 3
11	or	Logical Or	x == 2 or y == 3

Quiz: Bedingungen Teil 1 - Aufwärmen

Zu welchem Wert werden die Ausdrücke A bis D jeweils ausgewertet?

A
```
True and False
```

B
```
True or False
```

C
```
not True
```

D
```
10 == 12
```

Quiz: Bedingungen Teil 2 - Variablen

Welchen Wert hat die Variable k in den Skripten A und B jeweils?

A

```
x, y = 10, 3
k = x > y and y%2 == 0
```

B

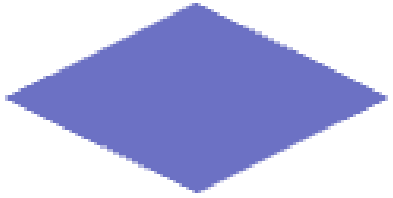
```
r, s = 100, "test"
k = s == "Test" or r >= 100
```

2.

Flussdiagramme

Etwas grafisches zu Beginn

Programm-Flussdiagramm



Branching

Start/Stop

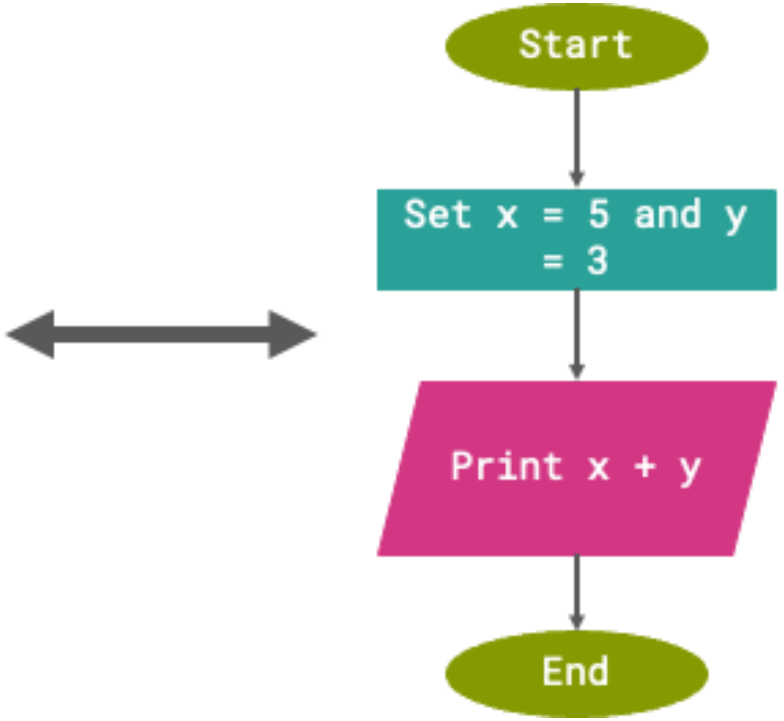
Operation /
Statement

Input/Output

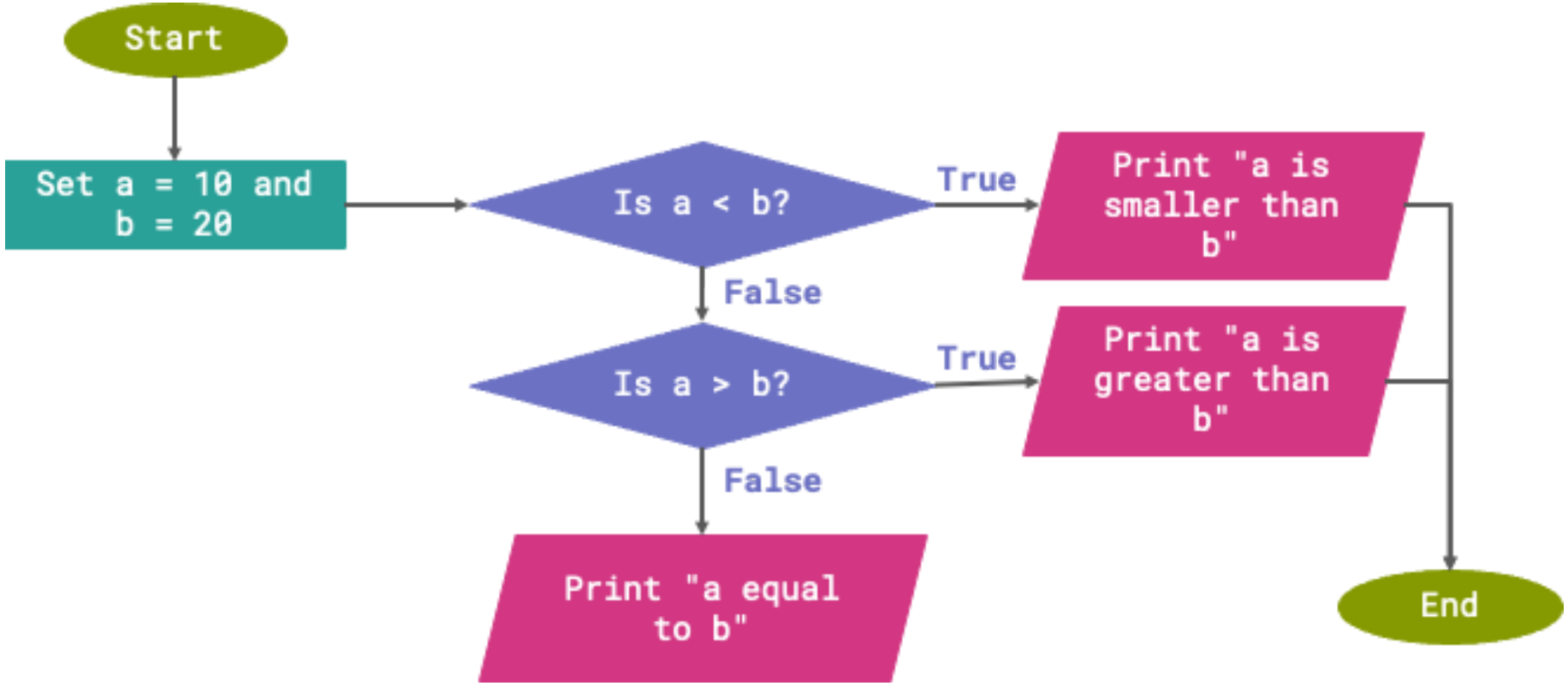
Programm-Flussdiagramm

```
x, y = 5, 3

print(x + y)
```



Bedingte Anweisungen



3.

if-elif-else

Bedingte Anweisungen

Bedingte Anweisungen

```
1  # 10-if-elif-else.py
2
3  a, b = 10, 20
4
5  if a < b:
6    print("a is smaller than b")
7  elif a > b:
8    print("a is greater than b")
9  else:
10   print("a is equal to b")
```

Bedingte Anweisungen

- if-Anweisung (mit Block) **`kann allein stehen`**
- **`elif- und else-Anweisungen müssen immer mit einer if-Anweisung verwendet werden`**
- **`keine oder eine beliebige Anzahl`** von elif-Anweisungen können verwendet werden
- die else-Anweisung ist **`optional`**

Bedingte Anweisungen

- Anweisungsblock: Ein zusammenhängender Abschnitt, der aus einer oder mehreren Anweisungen besteht
- Python verwendet Einrückung (Space- oder Tab-Taste) zur Definition eines Blocks

```
if a < b:
    print("a is smaller than b")
    print("Still belongs to Block 2")
else:
    print("This is a new block")
```

Quiz: Bedingte Anweisungen

- Was wird auf der Konsole ausgegeben, wenn der Code ausgeführt wird?

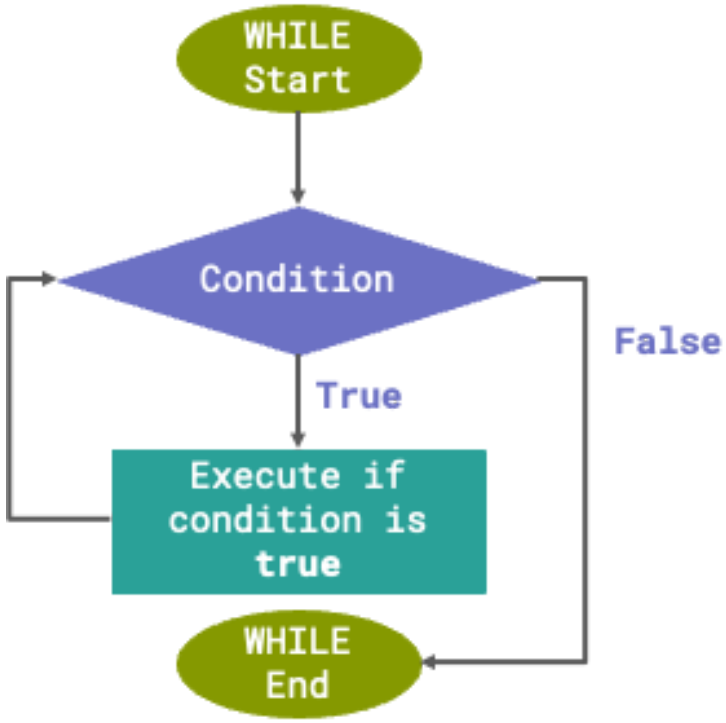
```
1 x = 100
2 if x > 0:
3   if x < 100:
4     print('A')
5   elif x > 30 and x % 2 == 0:
6     print('B')
7   else:
8     print('C')
9 print('D')
```

4.

while Schleife

Eine Erste Iteration

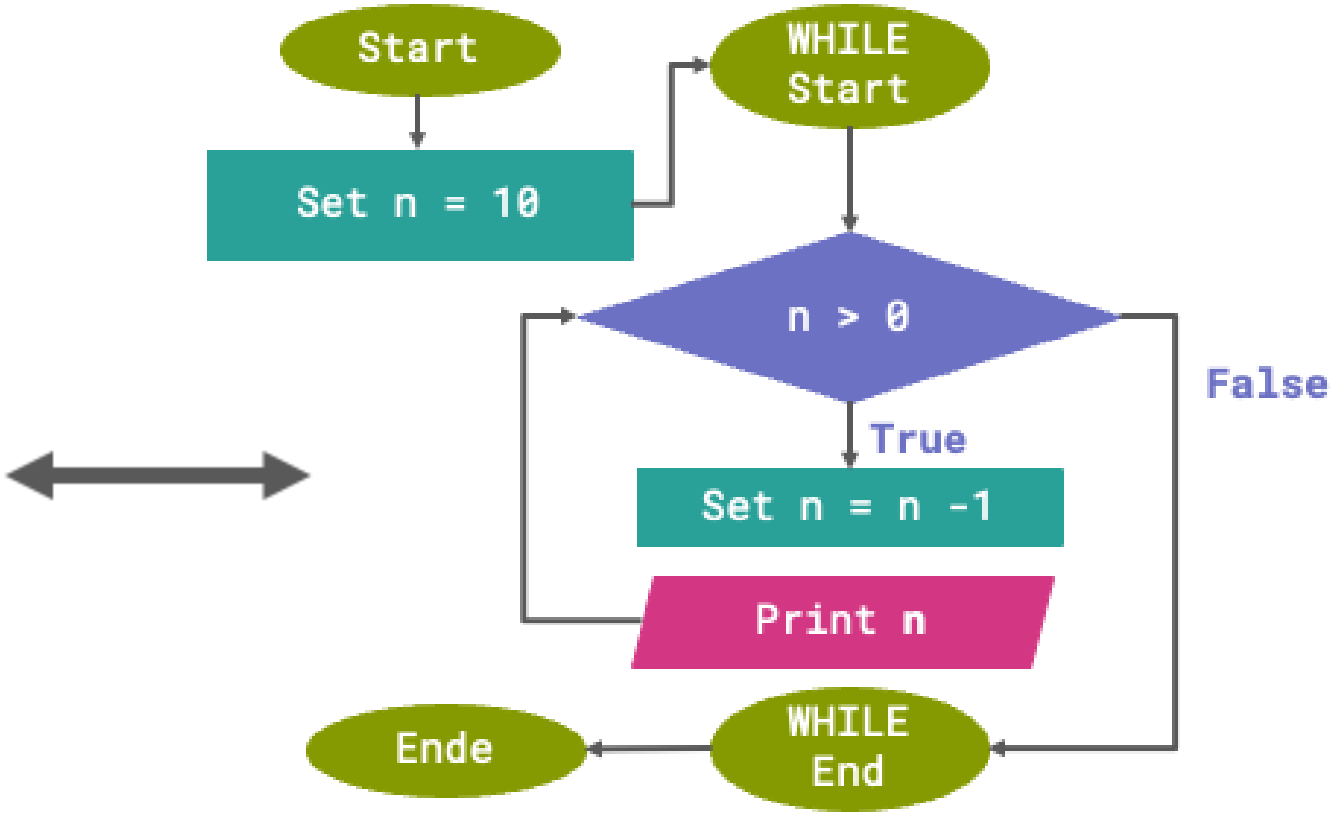
while Schleife




```
while <Condition>:
  <Execute if condition is true>
```

while Schleife

```
1  n = 10
2
3  while n > 0:
4      n = n -1
5      print(n)
```



break Ausdruck

```
1 # 10-while-break.py
2
3 n = 10
4 while True:
5   if n == 37:
6     print("Found the number 37")
7     break
8   n = n + 1
```

Der break Ausdruck kann nur innerhalb einer while- und/oder for-Schleife verwendet werden

Endlose Schleifen

```
1 # 10-endless-while.py
2
3 n = 0
4 while n < 10:
5   print("Hallo!")
```

Die Bedingung in der while-Schleife muss irgendwann zu **`False`** ausgewertet werden, oder die while-Schleife muss irgendwann durch **`break`** beendet werden, sonst gerät man in eine **`Endlosschleife`**

Quiz: while Schleife

Welche Zahlen bzw. wie viele werden ausgegeben, wenn der folgende Code ausgeführt wird?

```
1  n = 10
2
3  while True:
4    n = n - 1
5
6    if n == 1:
7      break
8
9    if n % 2 != 0 and n != 7:
10         print(n)
```

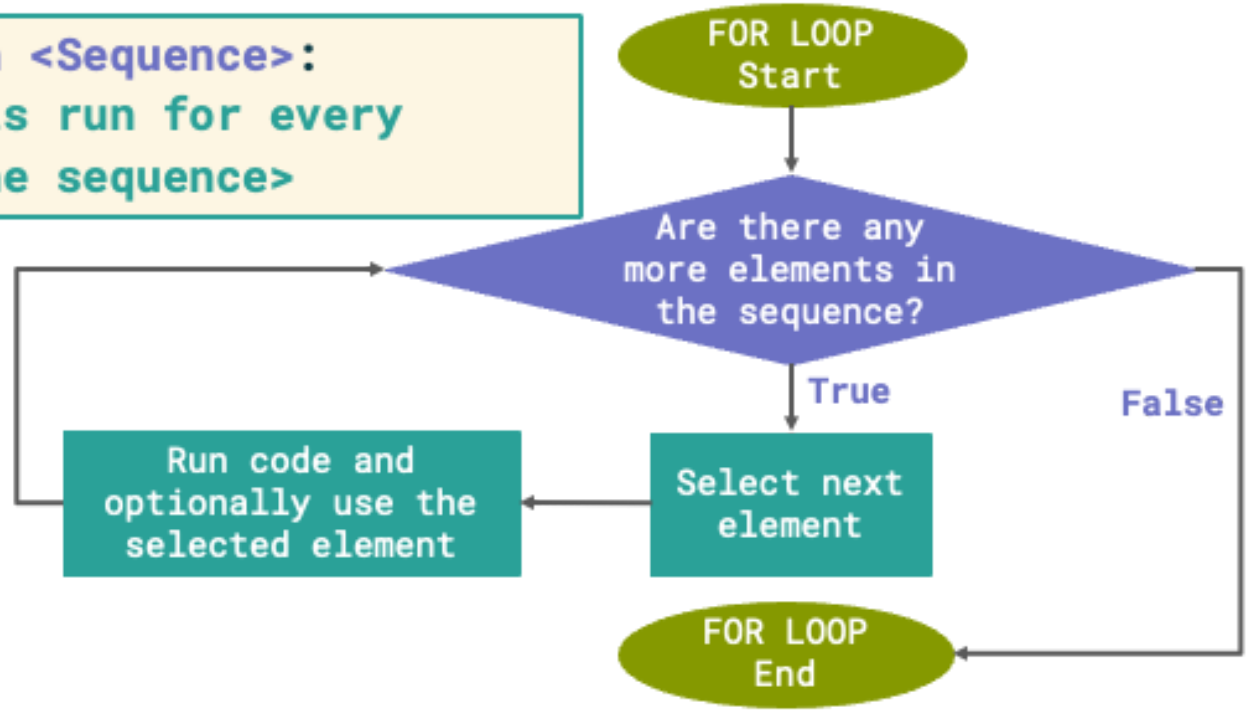
5.

for Schleife

Eine zweite Iteration

for Schleife

```
for <Element> in <Sequence>:
  <Code that gets run for every
   element in the sequence>
```



for Schleife

```
1 for n in [1, 2, 3]:
2   print(n)
```

Die Python for-Schleife wird immer auf Sequenzen oder andere iterierbare Objekte angewendet

Sequenzen: list, tuple und range

continue Ausdruck

```python
# 10-for-continue.py

for n in [1, 2, 3, 4, 5]:
  if n == 2:
    continue

  print(n)
```

Der continue Ausdruck kann nur innerhalb einer while- und/oder for-Schleife verwendet werden

Quiz: continue Ausdruck

```
1 for n in range(100):
2   if n%2 == 0:
3     continue
4
5   print(n)
```

Was bewirkt das obere Skript?

Übung: Einen Counter implementieren

Füllen Sie im folgenden Code die Lücken (___) so aus, dass die Zahlen von 1 bis einschliesslich 20 in aufsteigender Reihenfolge ausgegeben werden. Längere Lücken können aus mehreren Ausdrücken bestehen.

```
k = ___

while k _________:
  print(k)
  k = _______
```

Testen Sie Ihren Code anschliessend in PyCharm.

Lösung: Einen Counter implementieren

```
1 # 10-implement-a-counter
2
3 k = 1
4
5 while k <= 20:
6     print(k)
7     k = k + 1
```

Übung: Bedingungen definieren

Definieren Sie die folgenden Bedingungen als Python-Ausdrücke:

1. Die Zeichenfolge `s` muss mit "`Knb423Lo12`" übereinstimmen
2. `x` muss entweder grösser als 100 oder kleiner als 20 und durch 3 teilbar sein
3. `y` muss zwischen 10 und 20 liegen und darf nicht durch 7 teilbar sein

Lösung: Bedingungen definieren

```
# 10-define-conditions

s = "Knb423Lo12"
x = -12
y = 15

condition_1 = s == "Knb423Lo12"
condition_2 = (x > 100 or x < 20) and x % 3 == 0
condition_3 = (10 <= y <= 20) and y % 7 != 0

print(f'Is s == "Knb423Lo12"? {condition_1} (s has the value {s})')
print(f'Is the value of x either greater than 100 or less than 20 and divisible by 3?'
      f'{condition_2} (x has the value {x})')
print(f'Is the value of y between 10 and 20 and must not be divisible by 7?'
      f'{condition_3} (y has the value {y})')
```

Lösung: Grösste Nummer

```python
# 10-largest-number

nr_1 = int(input("Please enter a number (step 1 of 3): "))
# The first number we enter is initially the largest number,
# exactly until the moment when a larger number is entered
largest_nr = nr_1

nr_2 = int(input("Please enter a number (step 2 of 3): "))
if nr_2 > largest_nr:
    largest_nr = nr_2

nr_3 = int(input("Please enter a number (step 3 of 3): "))
if nr_3 > largest_nr:
    largest_nr = nr_3

print(f"The largest number you entered is {largest_nr}")
```

Übung: Grösste Nummer

Schreiben Sie ein Skript **`largest_number.py`**, das den Benutzer auffordert, drei ganze Zahlen einzugeben.

Nach der Eingabe der Zahlen sollte das Skript ausgeben, welche Zahl die größte ist.

Wenn der Benutzer zum Beispiel 100, 14 und 81 eingibt, sollte das Skript **`100`** auf der Konsole ausgeben.

Alternative Lösung: Grösste Nummer

```
1 # 10-largest-number
2
3 if nr_1 < nr_2:
4     if nr_2 < nr_3:
5         print(nr_3)
6     elif nr_2 > nr_3:
7         print(nr_2)
8 else:
9     print(nr_1)
```

Übung: Advanced Calculator

Schreiben Sie ein Skript **`advanced_calculator.py`**, das zwei Ganzzahlen und einen Operator (+, -, / oder *) einliest.

Geben Sie das Ergebnis je nach Operator auf dem Bildschirm aus. Wenn der eingegebene Operator nicht unterstützt wird, geben Sie stattdessen eine **`Fehlermeldung`** aus.

Lösung: Advanced Calculator

```python
# 10-advanced-calculator

nr_1 = int(input("Please enter a number (step 1 of 2): "))
nr_2 = int(input("Please enter a number (step 2 of 2): "))
operator = input("Please select an operator (+, -, /, or * are allowed): ")

if operator == "+":
    print(f"The result is {nr_1 + nr_2}")
elif operator == "-":
    print(f"The result is {nr_1 - nr_2}")
elif operator == "/":
    print(f"The result is {nr_1 / nr_2}")
elif operator == "*":
    print(f"The result is {nr_1 * nr_2}")
else:
    print("Error: Invalid operator was selected!")
```

Ende

Das war alles für dieses Kapitel