3. Schleifen

Wie bereits mit Verzweigungen im Kapitel „2. Datentypen und Verzweigungen“, gezeigt, kann ein Programm aufgrund einer Bedingung bestimmte Anweisungen ausführen und manche nicht. So zeigte sich, dass Programme nicht nur Zeilenweise von oben nach unten abgearbeitet werden.

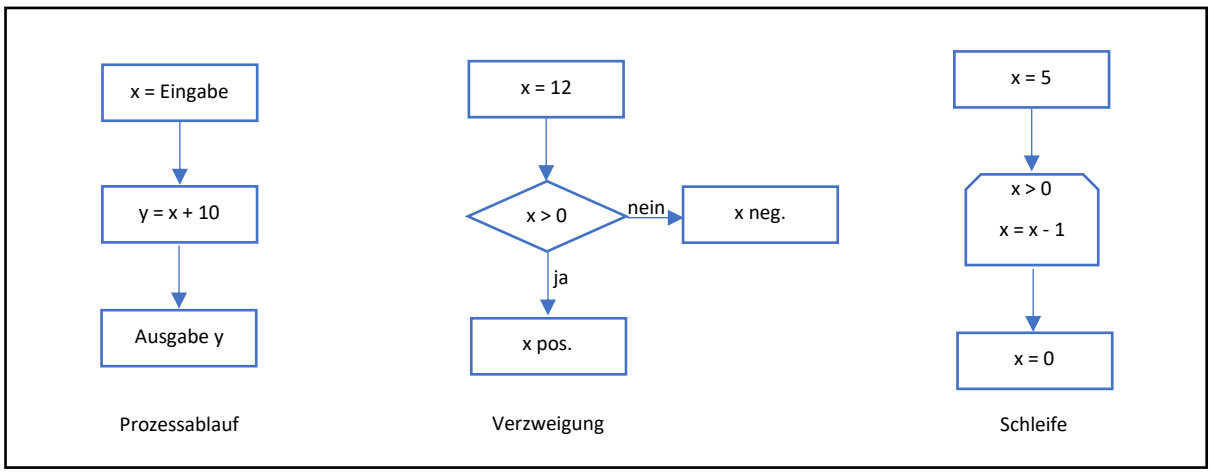

Abb.11: Programmablaufplan (=PAP): Prozessablauf, Verzweigungen und Schleifen

Ein Programm kann eine Prozessablaufrichtung haben, dabei werden Prozesse nacheinander ausgeführt. Ab einem Punkt im Programm kann der Prozess in **mehr als eine Ablauflinie** aufgeteilt (Verzweigung) oder **als Zyklus wiederholt** (Schleife) werden.

Mit den **Schleifen** können nun Anweisungen bzw. Prozesse wiederholt ausgeführt werden. Es gibt zwei Arten von Wiederholungen (Iterationen):

a.) Zählschleife (spezielle Iteration) **for** – Schleife
b.) bedingungsgesteuerte Schleife (bedingte Iteration) **while** - Schleife

for – Schleife

```
1  for i in (1, 2, 3, 4, 5):
2      print(i)
3  print("Ende des Programms")
```

Ausgabe: 1 2 3 4 5 Ende des Programms

while – Schleife

```
1  Ztimestart=10
2  while Ztimestart > 0:
3      print(Ztimestart)
4      Ztimestart = Ztimestart – 1
5  print(“Rakete startet…“)
```

Ausgabe: 10 9 8 7 6 5 4 3 2 1 Rakete startet…

Ein **vorzeitiger Abbruch** in einer Schleife kann durch den Befehl **break** in Kombination mit einer Bedingung erzwungen werden.

Erstellen wir nun ein Python-Programm, wie bei dem Beispiel der **while**-Schleife und nutzen den Befehl **break**. Nach „5“ soll es noch aus sicherheits- bzw. technischen Gründen möglich sein den Count-Down des Raketenstarts abzubrechen. Falls der Check mit „j“ beantwortet wird startet die Rakete, ansonsten wird der Start der Rakete abgebrochen.

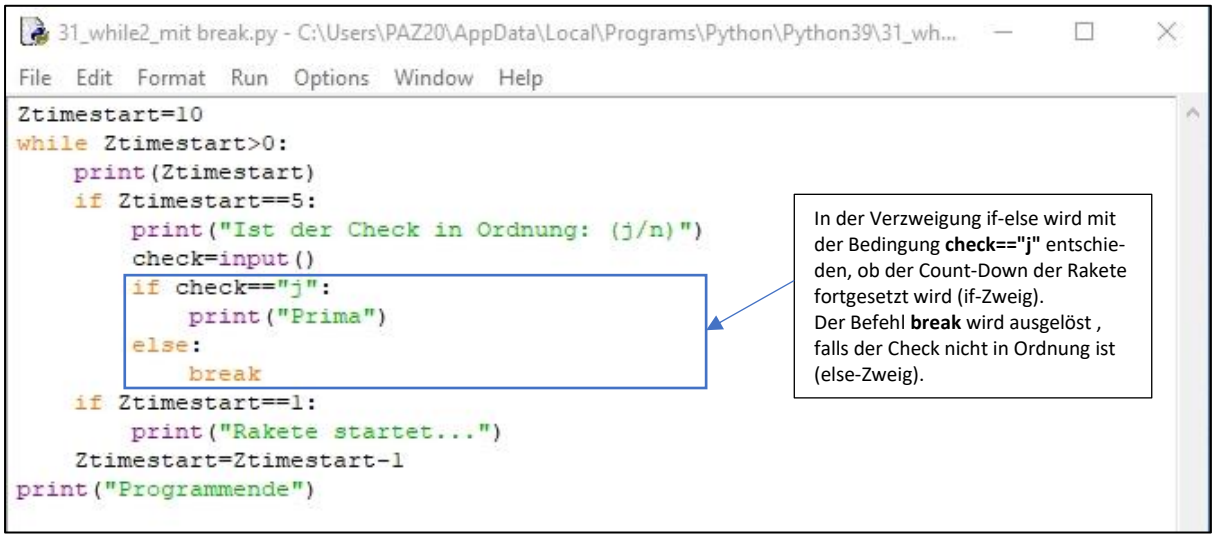

Abb.12: Python-Programm: Rakete startet nach erfolgreichem Check

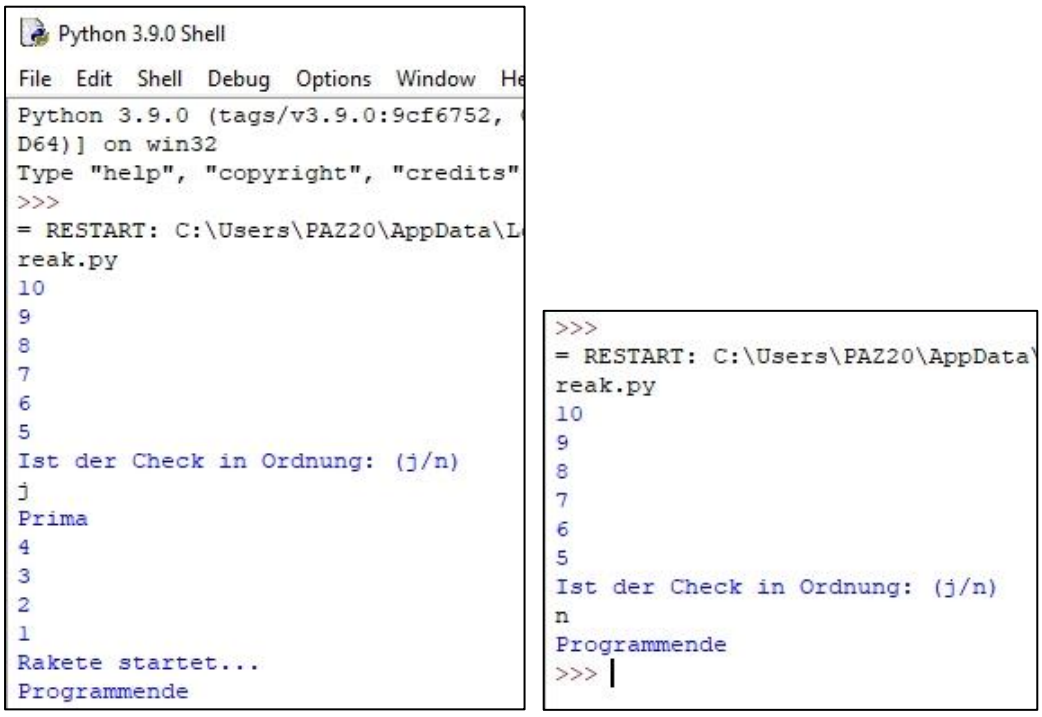

Abb.13: Python-Shell: Rakete startet nach erfolgreichem Check

Der Befehl **continue** sorgt dafür, dass man den Rest einer Schleife überspringt, um dann die Anweisungen aus der nächsten Runde zu wiederholen (üblicherweise die nächste "Iteration" genannt).

Erstellen wir nun ein Python-Programm, wie bei dem Beispiel der **for**-Schleife und nutzen den Befehl **continue**. Die Zahl 13 ist eine Unglückszahl, diese soll beim Abzählen nicht ausgegeben werden. Nutzen Sie die **range()**-Funktion.

Statt die Zahlen aufzuzählen wird oft bei der for-Schleife die **range()** Funktion angewendet. Bei der Funktion **range(b,e)** kann der Bereich einer Zahlenfolge mit zwei Argumenten angegeben werden: **b** = Beginn des Bereichs **e** = Ende des Bereichs

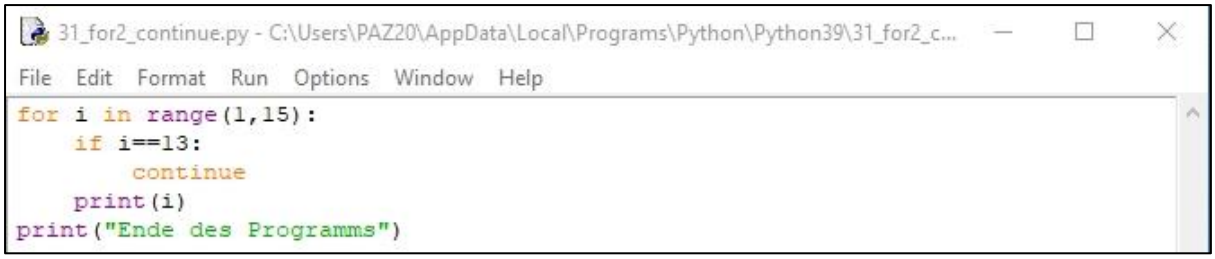

Abb.14: Python-Programm: Zahlen von 1 bis 15 ausgeben, ohne der Unglückszahl 13

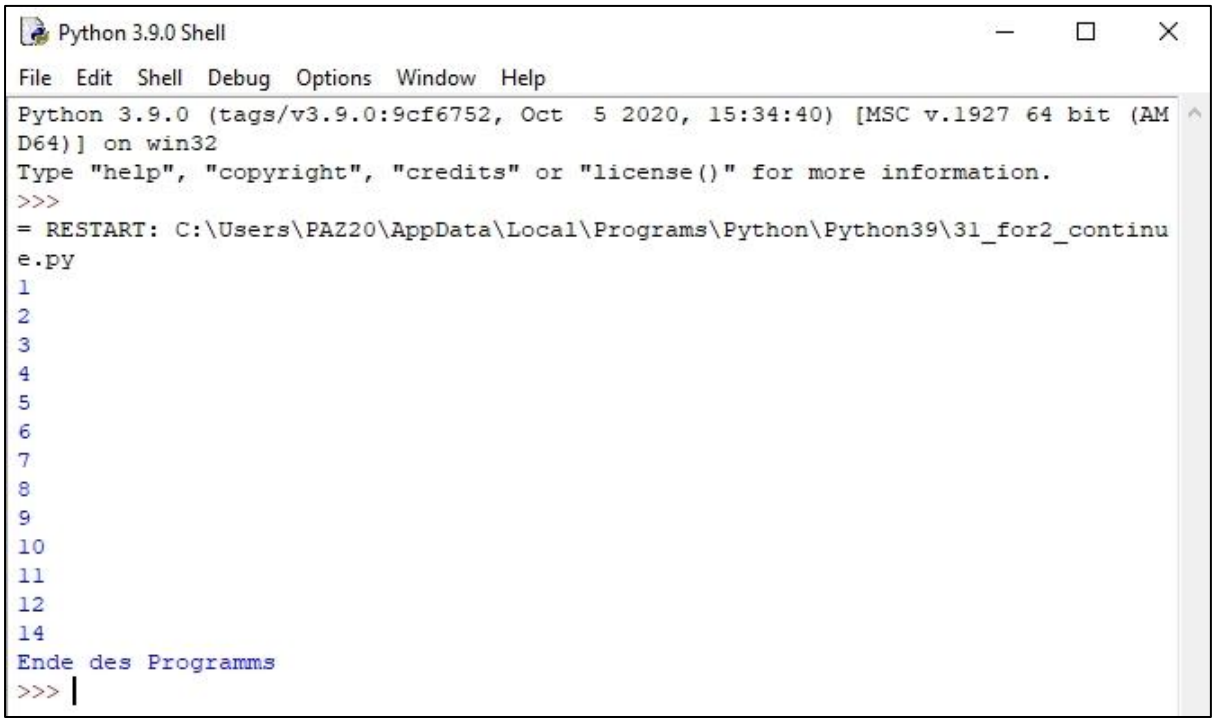

Abb.15: Python-Shell: Zahlen von 1 bis 15 ausgeben, ohne der Unglückszahl 13

Fehler und Ausnahmen

Bei komplexeren Programmen ist es wichtig auch die **Fehler** nach einer Eingabeaufforderung (z.B. Anwender gibt keine Zahl ein, sondern einen Text) abzufangen, sonst wird das Programm mit einer Fehlermeldung beendet.

Welche kritischen Stellen im Programm gibt es?

a.) Eingabeaufforderung (die nicht richtig ist)
b.) Bearbeitung einer Datei (die nicht existiert)
c.) Ausgabe an einen Drucker (der nicht eingeschaltet ist)

Die Ausnahmebehandlung, z.B. wo die Eingabeaufforderung erfolgt, wir mit **try** eingeleitet. Ähnlich wie bei if-Anweisung werden die eingerückten Anweisungen nach **try** ausgeführt try-Zweig. Falls die Eingabe erfolgreich war, wird der except-Zweig nicht ausgeführt, ähnlich wie bei else-Zweig der If-Anweisung.

Ist die Eingabe dagegen nicht erfolgreich und handelt es sich um einen Fehler, wird der Fehler oder die Ausnahme mit der Anweisung **except** abgefangen. Alle eingerückten Anweisungen nach dem except-Zweig werden dann ausgeführt. Das Programm läuft ohne Abbruch zu Ende.

Nach **try** und **except** muss jeweils ein Doppelpunkt gesetzt werden, wie bei if-else, for oder while.

1	fehler = 1	
2	while fehler==1:	
3	print("Bitte geben Sie eine ganze Zahl ein")	
4	z = input()	
5	try:	
6	zahl = int(z)	
7	print("Eingabe korrekt: ", zahl)	← try-Zweig
8	fehler = 0	
9	except:	
10	print("Eingabe nicht korrekt. ")	← except-Zweig
11	print("Ende des Programms")	

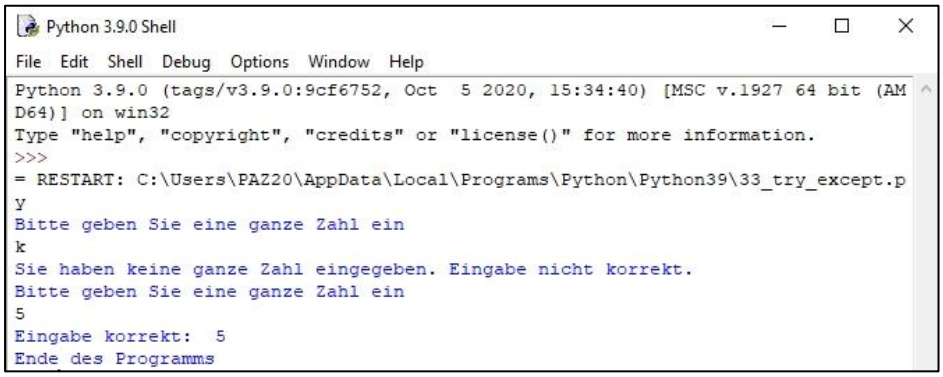

Abb.16: Python-Shell: Falls die Eingabe nicht korrekt ist, wird das Programm erneut ausgeführt