

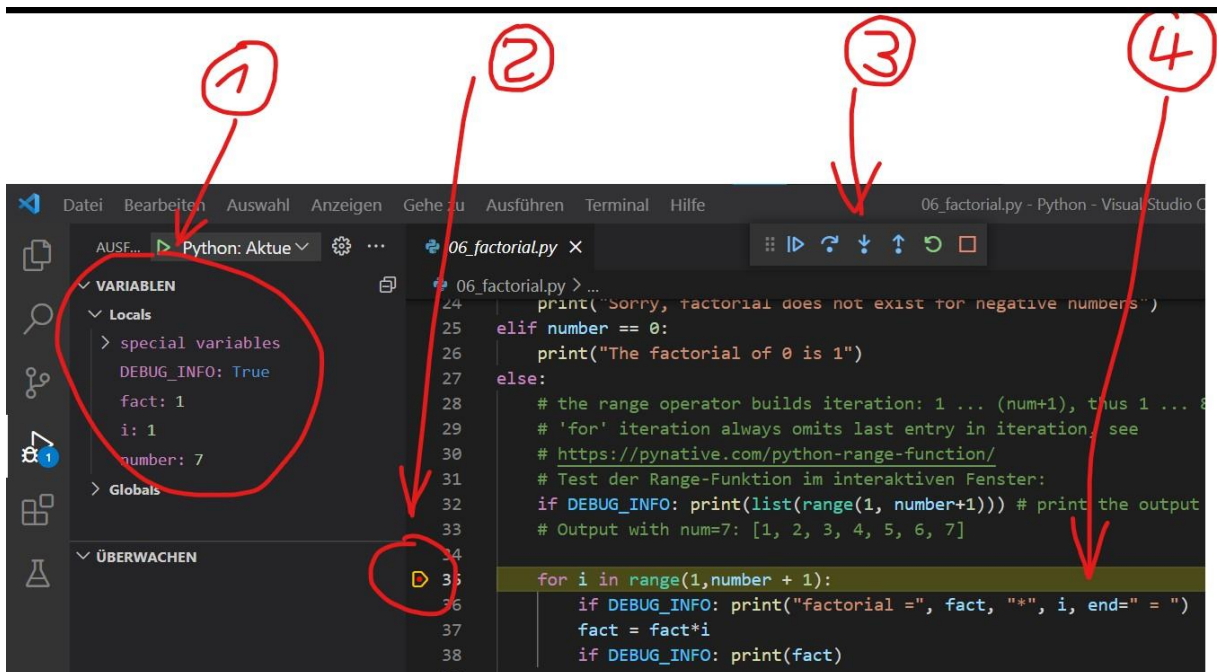
Debugging

Am Beispiel von factorial.py (Berechnen der Fakultät einer Zahl)

Um das Debugging zu starten, setzen wir zunächst einen Haltepunkt („Breakpoint“) an beliebiger Stelle im Programmcode. Hierzu klicken wir in die Spalte rechts neben der Zeilennummerierung. Der Breakpoint wird als roter Punkt dargestellt. Als Breakpoint empfiehlt sich häufig der Beginn oder eine Zeile innerhalb einer (for- bzw. while-) Schleife.

Danach starten wir den Debugger mit „Ausführen > Debugging starten (F5)“.

Das Programm endet an der Codezeile, an der zuvor der Breakpoint gesetzt wurde (s. Abb. 1, Punkt 2).



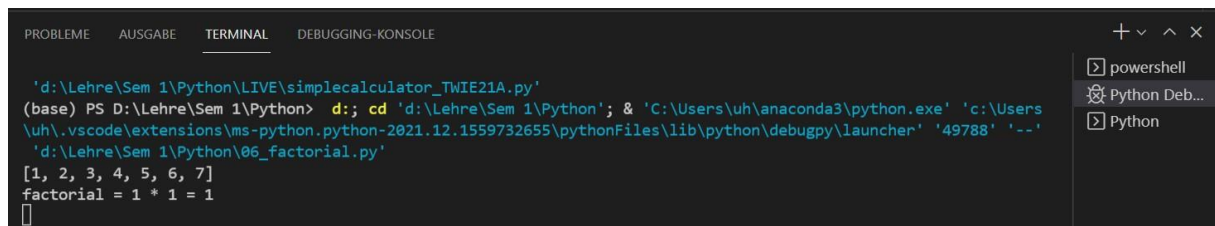
Die Codezeile wird zudem gelb eingefärbt.

Mit Hilfe der Überwachungsfenster auf der linken Seitenhälfte lassen sich sämtliche lokale und globale Variablen, nur eigene Variablen sowie die Aufrufliste der beteiligten Programme überwachen (Abb 1, Punkt 1).

Mit Hilfe der Schaltflächen in Abb 1, Punkt 3 lässt sich das Programm bis zum Ende fortsetzen („Weiter“), in Prozedur- oder Einzelschritten fortsetzen, zurückspringen, neu starten und stoppen.

Zudem hat der Programmierer die Möglichkeit, alternativ oder als Ergänzung zum Debugging, dezidierte Debug-Print-Zeilen dem Programm hinzuzufügen (s. Abb. 1, Punkt 4). Diese wollten nur dann ausgeführt werden, wenn DEBUG_INFO zuvor auf True gesetzt wurde.

Auf dem Terminalfenster können sämtliche (Debug-)Print-Ausgaben verfolgt werden (s. Abb. 2).



```
PROBLEME  AUSGABE  TERMINAL  DEBUGGING-KONSOLE

'd:\Lehre\Sem 1\Python\LIVE\simplecalculator_TWIE21A.py'
(base) PS D:\Lehre\Sem 1\Python> d:; cd 'd:\Lehre\Sem 1\Python'; & 'C:\Users\uh\anaconda3\python.exe' 'c:\Users\uh\.vscode\extensions\ms-python.python-2021.12.1559732655\pythonFiles\lib\python\debugpy\launcher' '49788' '--'
'd:\Lehre\Sem 1\Python\06_factorial.py'
[1, 2, 3, 4, 5, 6, 7]
factorial = 1 * 1 = 1
█
```

Ziel des Debuggings ist es, insbesondere die Variablen, welche innerhalb einer Schleife verändert werden, zur Laufzeit zu beobachten. Fehler, die während der Laufzeit entstehen, können so leicht entdeckt und nach dem Debuggen korrigiert werden.

So ist ein typischer Fehler im oberen Beispiel, den range-Operator falsch zu konfigurieren, z.B. `range(1, numer)` statt `range(1, number + 1)`. Durch das Debugging wird dieser Fehler schnell erkannt!