Arbeitsauftrag GK inf12 für 24.11.2020

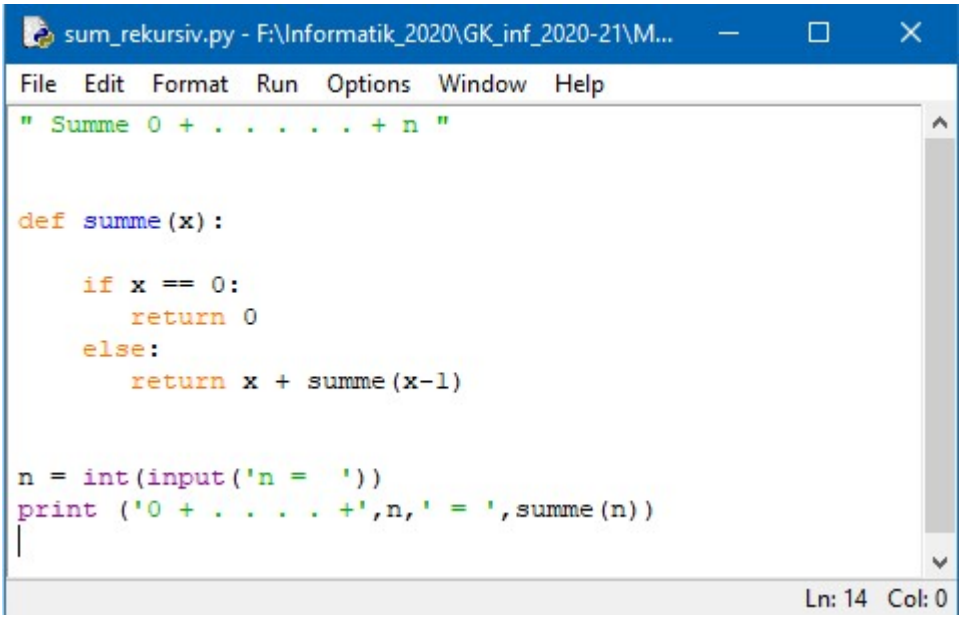
In höheren Programmiersprachen (wie Python) ist die Möglichkeit implementiert, Funktionen, auch rekursiv formulierte Funktionen, zu definieren. Dabei verstehen wir unter einer Funktion ein Unterprogramm (Prozedur), welches nach der Übergabe von Daten einen Funktionswert an das aufrufende Programm zurückgibt. Der zur Funktion gehörende Anweisungsblock heißt auch Funktionsrumpf (in Python wird der Funktionsrumpf durch Einrücken des Programmtextes kenntlich gemacht). Eine Funktion, deren Funktionsrumpf mindestens einen Aufruf ihrer selbst enthält, heißt rekursiv (lat. recurrere, zurücklaufen).

Die Funktion **summe** (siehe Arbeitsauftrag für 17.11.2020), die einer natürlichen Zahl **n** mit n ∈ {0, 1, 2, . . . } die Summe
0 + ... + n zuordnet, läßt sich rekursiv wie folgt definieren:

Rekursionsanfang: **summe(0) = 0**

Rekursionsvorschrift: **summe(n) = n + summe(n – 1)** falls n ≥ 1

Realisierung von **summe** in Python:



Erläuterungen:

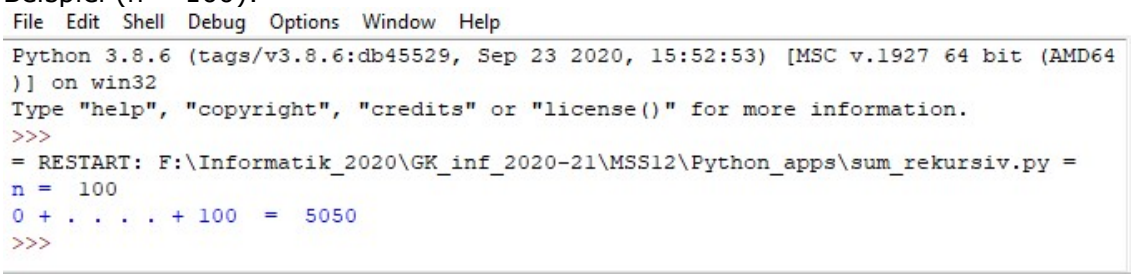
def summe(x): Funktionskopf;
summe = Name der Funktion
x = lokale (nur innerhalb der Funktion verfügbare) Variable
Nach dem Doppelpunkt folgt der durch Einrücken kenntlich gemachte Funktionsrumpf.

return mit **return** wird der berechnete Funktionswert an das aufrufende Programm übergeben

Der Aufruf **summe(n)** (hier: innerhalb der **print**-Anweisung) bewirkt:

- Der aktuelle Wert der Variablen **n** wird der lokalen, nur innerhalb der Funktion verfügbaren Variablen **x** zugewiesen
- Nach der (hier rekursiv erfolgenden) Berechnung wird der Funktionswert mit **return** zurückgegeben.

Beispiel (n = 100):



```
File  Edit  Shell  Debug  Options  Window  Help
Python 3.8.6 (tags/v3.8.6:db45529, Sep 23 2020, 15:52:53) [MSC v.1927 64 bit (AMD64
)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: F:\Informatik_2020\GK_inf_2020-21\MSS12\Python_apps\sum_rekursiv.py =
n =  100
0 + . . . . + 100  =  5050
>>>
```

Die rekursive Berechnung von **summe(6) = s(6)** läßt sich wie folgt verdeutlichen:

```
s(6) = 6 + s(5)
     = 6 + (5 + s(4))
     = 6 + (5 + (4 + s(3)))
     = 6 + (5 + (4 + (3 + s(2))))
     = 6 + (5 + (4 + (3 + (2 + s(1)))))
     = 6 + (5 + (4 + (3 + (2 + (1 + s(0))))))
```

mit **s(0)** ist der Rekursionsanfang erreicht, die Rekursion bricht ab.

Arbeitsaufträge für 24.11.2020:

1.) Erstelle den Programmtext für die rekursive Berechnung von **summe(n)**; man orientiere sich an dem obenstehenden screenshot.

2.) Teste das Programm sowohl als iterativ (gemäß Ziffer 1 aus Arbeitsauftrag für 17.11.2020) als auch als rekursiv definierten Algorithmus für unterschiedliche Werte von n; wähle auch n = 1000, 10000, 100000, 1000000. Was fällt auf?

3.) Die Fakultätsfunktion (eng.: factorial) läßt sich rekursiv definieren (vgl. das Paper „Funktionaler_und_Imperativer_Ansatz" vom 27.10.2020):

Rekursionsanfang: **fact(0) = 1**

Rekursionsvorschrift: **fact(n) = n · fact(n – 1)** , falls n > 0

Schreibe und teste ein Python-Programm, um die Fakultätsfunktion rekursiv zu berechnen; vergleiche mit dem iterativ formulierten Algorithmus gemäß Ziffer 2 des Arbeitsauftrags für 17.11.2020.

4.) fakultativ: Aufgabe Nr. 5 aus Aufgabenblatt Nr. 2 vom 10.11.2020