SOURCE CODE 1. Erzeugung eines kartesischen Produktes in lexikographischer Ordnung

```
def product_lex(maxima):
    """Erzeuge das kartesische Produkt
    {1,...,maxima[0]} x {1,...,maxima[1]} x ..."""
    # Anzahl der Faktoren des kartesischen Produkts:
    dim = len(maxima)
    # Lexikographisch kleinstes Element in diesem
    # kartesischen Produkt ist (mathematisch) das Tupel (1,1, ..., 1),
    # in Python die Liste [1,1,...,1] (list und tuple sind in Python
    # _verschiedene_ Datentypen!):
    current_tuple = [1]*dim
    # deepcopy verwenden: Erzeuge eine KOPIE von current_tuple
    # und gib diese als erstes Element in die Liste von Listen,
    # die in der Folge zum gesamten kartesischen Produkt erweitert
    # wird:
    cart_prod = [ deepcopy(current_tuple) ]

    def find_last_pos(n,ct,tm):
        """Hilfsfunktion innerhalb der Funktion product_lex:
        Sie soll aufgerufen werden mit n=dim, ct=current_tuple
        und tm=the_maxima."""
        # Suche die "am weitesten rechts stehende" Komponente
        # des Tupels, die kleiner ist als das Maximum für diese
        # Komponente (die also "hochgezählt" werden kann):
        for i in range(n-1,-1,-1):
            # Range von n-1 bis 0 in absteigender Reihenfolge
            if ct[i] < tm[i]:
                return i
        return -1

    # Der "walrus-operator := " liefert als _Ergebnis_ den zugewiesenen
    # Wert (d.h.: Das, was rechts von ":=" steht): Damit können wir die
    # Abbruchbedingung für die while-Schleife so schreiben:
    while (last_pos := find_last_pos(dim, current_tuple, maxima)) > -1:
        # Komponente last_pos kann "hochgezählt" werden ...
        current_tuple[last_pos] += 1
        # ... und alle Komponenten "links von last_pos" werden auf 1 gesetzt:
        current_tuple[last_pos+1:] = [1]*(dim-last_pos-1)
        # Nicht vergessen: KOPIE erzeugen mit deepcopy!
        cart_prod.append(deepcopy(current_tuple))

    # Wenn wir damit fertig sind, geben wir die erzeugte
    # Liste zurück:
    return cart_prod
```

An diesem einfachen Beispiel illustrieren wir zugleich einige Feinheiten von Python: Darum sind die Kommentare viel umfangreicher als die eigentlichen Programmzeilen.