

SOURCE CODE 1. Generator: Gray–Code für cartesisches Produkt.

```

def product_gray_code(the_maxima):
    """Erzeuge Gray-Code des cartesischen Produkts: Diesmal
    beginnen wir mit dem Zählen gleich bei Null, betrachten also
    {0,...,m_1-1} x {0,...,m_2-1} x ... x {0,...,m_n-1}.
    ACHTUNG: Hier muß min(the_maxima) > 1 gelten! (Faktoren der
    Mächtigkeit 1 sind ja "in allen Tupeln konstant", und
    Faktoren der Mächtigkeit 0 ergeben ein leeres Produkt;
    das ist also keine starke Einschränkung.)
    """
    if min(the_maxima) <= 1:
        # Diesen Fall behandeln wir hier nicht.
        print('Wir behandeln nur Produkte von Faktoren S mit |S|>1!')
        return

    dim = len(the_maxima)
    # Erstes Tupel:
    current_tuple = np.zeros(dim, dtype=int)
    # Vektor, der die "Richtungen" anzeigt, in die
    # die Komponenten des Tupels während des Algorithmus
    # bewegt werden:
    directions = np.ones(dim, dtype=int)
    # Vektor, der die "aktiven" Koordinaten anzeigt:
    active_flags = np.ones(dim, dtype=int)

    finished = False
    while not finished:
        yield current_tuple
        # Die numpy-Funktion nonzero(v) liefert array der Indices
        # der Elemente von v, die ungleich Null sind; genauer gesagt:
        # Ein _Tupel_ von arrays von Indizes - für jede Dimension
        # von v ein Index-Array; unser v is eindimensional, wir brauchen
        # also einfach das ERSTE Element (mit Index 0) dieses Tupels:
        active_positions = np.nonzero(active_flags)[0]
        # Wenn es keine aktiven Koordinaten mehr gibt: Fertig!
        if len(active_positions) == 0:
            return
        # Die größte aktive Koordinate wird in die entsprechende
        # Richtung (+/- 1) verändert:
        pos = np.max(active_positions)
        current_tuple[pos] += directions[pos]
        # Wenn wir an die "oberen/unteren Grenzen" gestoßen sind,
        # kehren wir die Richtung um und setzen die Koordinate auf
        # "nicht aktiv":
        if current_tuple[pos]==0 or current_tuple[pos]==the_maxima[pos]-1:
            directions[pos] = -directions[pos]
            active_flags[pos] = 0
        # Setze alle active-flags ab pos+1 auf 1: Die entsprechenden
        # Koordinaten können im nächsten Schleifen-Durchlauf verändert

```

werden. - Zuweisung einer Zahl zu numpy-array-slice ist möglich!

```
active_flags[pos+1:] = 1
```