

## Austabellieren von der Funktion $p(n,k)$ mithilfe einer Rekursion.

```
def p_nk_by_recursion(nn):
    """Erzeuge eine Tabelle der Zahlen  $p(n,k)$  für  $0 \leq n, k \leq nn$ """
    pnk = np.zeros((nn+1)**2, dtype=int).reshape((nn+1, nn+1))
    # Fülle Zeile 0 mit Einsern: Es gibt _eine_ Partition von 0,
    # nämlich die _leere_ Partition.
    pnk[0][:] = 1
    # Fülle Zeile 1 ab Spalte 1 mit Einsern: Für  $n=1$  gibt es
    # nur eine Partition, und deren erster Teil ist  $1 > 0$ .
    pnk[1][1:] = 1
    # Wir bestimmen alle Partitionen von  $n$  ...
    for n in range(2, nn+1):
        # Mit größtem (also ersten) Teil kleiner gleich  $k$  ...
        for k in range(1, n+1):
            # Durch eine Rekursion:
            for j in range(1, k+1):
                for i in range(1, n//j+1):
                    pnk[n][k] += pnk[n-j*i, j-1]
        for k in range(n+1, nn+1):
            pnk[n, k] = pnk[n, n]
    return pnk
```

Hier wird die Rekursion  $p(n,k) = \sum_{j=1}^k \sum_{i=1}^{[n/j]} p(n - i \cdot j, j-1)$  verwendet; die Ergebnisse werden wieder in einem numpy-Array gespeichert: Für größere  $n$  ist das Verfahren deutlich schneller als die "Brute-Force-Abzählung" aller Partitionen, die man mit `generate_integer_partitions` erzeugt.