Source Code 1. Erzeugung zufälliger Zahl–Partitionen.

```
def random_integer_partitions(n):
    """Generator, der einen Iterator liefert, der zufällig
    gewählte Partitionen von n erzeugt"""
    pnk = p_nk_by_recursion(n)
    p_n = pnk[n][n]
    print(f'Es gibt {p_n} Zahlpartitionen von {n}.')
    while True:
        # Wähle zufällig rank in [0, p_n-1] ...
        rank = np.random.randint(0, p_n)
        # ... und erzeuge die Zahlpartition
        # mit diesem Rang:
        pi = []
        num = p_n - rank - 1
        n_j = n
        while n_j:
            pi_j = np.count_nonzero(pnk[n_j] <= num)
            if pi_j == 1:
                pi+= [1]*n_j
                break
            num-=  pnk[n_j][pi_j - 1]
            pi+=   [pi_j]
            n_j-=  pi_j
        # Befehl 'yield' macht aus der Funktion einen Generator:
        yield np.array(pi)
```

Wir verwenden die Bausteine zum Thema Zahl–Partitionen für einen "Zufalls–Zahlpartitionen–Generator": Die Tabelle mit den rekursiv berechneten Werten $p(n,k)$ berechnen wir hier natürlich nur *einmal*.