Source Code 1. Experimentelle Mathematik mit einer Monte–Carlo–Simulation.

```python
def monte_carlo_permutations(n, sample_size, test_permutation):
    """Erzeuge zufällig sample_size Permutationen von {1,2,...,n}
    und liefere die relative Häufigkeit der Permutationen, für die
    die Testfunktion test_permutation den Wert True liefert."""
    nfact = np.math.factorial(n)
    print(f'MC-simulation: Generate {sample_size} random permutations')
    print(f'of {n} elements ({n}! = {nfact}) and count for how many')
    print(f'of them function "{test_permutation.__name__}" returns True:')
    # Erzeuge einen Zufallszahlen-Generator (Funktionalität der
    # Numerik-Bibliothek numpy)
    random_number_generator = np.random.default_rng()
    # Der Zufallszahlen-Generator erzeugt sample_size zufällig
    # gewählte ganze Zahlen im Intervall von 0 (inklusive) bis n!
    # (exklusive); in Form eines numpy-Arrays.
    random_integers = random_number_generator.integers(
        low=0,
        high=np.math.factorial(n),
        size=sample_size
    )
    count = 0
    for i in random_integers:
        if test_permutation(unrank_perm_lex(i, n)):
            count+= 1
    return count/sample_size
```

Einfaches Beispiel für eine Monte—Carlo–Simulation: Erzeuge zufällig Permutationen von $[n]$ und zähle, für wieviele davon die Testfunktion `test_permutation` den Wert True liefert. Zugleich ein kleines Beispiel dafür, wie das Formatieren von Strings in Python funktioniert. (Es gibt aber *mehrere* Arten, Strings zu formatieren!)